Volumen

XXXI Conferencia Latinoamericana de Informática

Pontificia Universidad JAVERIANA Cali

Universidad del Valle

ParqueSoft®

Obras Completas CLEI2005

Volumen

XXXI Conferencia Latinoamericana de Informática

Editores
Juan Francisco Díaz
Camilo Rueda
Antal A. Buss

Octubre 10-14 de 2005, Cali, Colombia
http://www.clei2005.org

Obras Completas CLEI2005
Volumen: XXXI Conferencia Latinoamericana de Informática

CD: ISBN 958-670-422-X (Obra completa)
Libro: ISBN 958-670-423-8

Contenido/Contents

Prólogo ii

Comité de Programa iii
XXXI Conferencia Latinoamericana de Informática . . . iii
XII Concurso Latinoamericano de Tesis de Maestría . . . v
Comité Organizador CLEI2005 . . . vi

Artículos XXXI Conferencia Latinoamericana de Informática 1
A New Similarity Measure for Comparing
Gonzalo Müller; Claudio Rocco . . . 3
Invariant Pattern Recognition by Projection Histograms and Fuzzy ART neural network
Guillermo Cámara Chávez; Arnaldo de Albuquerque Araújo . . . 11
Constraint Solving Using Dynamic Variable and Value Selection
Carlos Castro; Christian Figueroa; Eric Monfroy; Rafael Meneses . . . 19
Estudio Experimental de un Algoritmo rápido de Intersección para Secuencias Ordenadas
Ricardo Baeza-Yates; Alejandro Salinger . . . 29
A Modification of the Landau-Vishkin Algorithm Computing Longest Common Extensions via Suffix Arrays
Rodrigo Miranda; Mauricio Ayala-Rincon . . . 41
Uso de Funciones de Base Radial en Monitoreo Ambiental
Cristian Rusu; Virginia Rusu . . . 51
Sistema experto difuso para la determinación de estrategias de contratación en agentes comercializadores de energía eléctrica en Colombia
Julián Moreno Cadavid; Demetrio Arturo Ovalle Carranza . . . 59
Sistema para detecção da borda da pupila do olho humano a partir de foto digital.
Elisamara de Oliveira; Patrícia Targon Campana; César Augusto Cardoso Caetano; Sandro Aparecido Ferraz . . . 73
CoGrOO - Corretor Gramatical para a língua portuguesa, acoplável ao OpenOffice
Jorge Kinoshita; Carlos Eduardo Dantas de Menezes; Laís do Nascimento Salvador . . . 79
Identificação de Spam através de uma Rede Neural Backpropagation
Ana C. Bittencourt; Silvia M.W. Moraes . . . 87
Aplicación de Tecnología de Agentes Asistentes de Usuario para la Resolución de Consultas en Repositorios de Datos Genómicos
David Benaderet; Ernesto Ocampo; Ana Janauskas . . . 97
Uma Ferramenta para Converter Exemplos de Aprendizado do Formato Atributo-Valor para o Formato Apropriado ao Uso de Algoritmos de Programação Lógica Indutiva
Mariza Ferro; Maria Carolina Monard . . . 109
Epsilon-PSO: A Particle Swarm Optimizer with Epsilon-Greedy Action Selection
Jorge Peña; Roberto Hincapié; Jorge Barrera . . . 121
A Alternativa Construtivista em Inteligência Artificial
Filipo Perotto; Luís Otávio Alvares . . . 129
Intelligent e-learning platforms infrastructure
Marta Zorrilla; Socorro Millán; Ernestina Menasalvas . . . 141
Uma Proposta para a Construção de Ensembles Simbólicos que Explicam suas Decisões
Flavia Cristina Bernardini; Maria Carolina Monard . . . 151
Um Proposta de Novo Método de Seleção Roleta para os Algoritmos Genéticos Baseado no Paradigma do Dilema do Prisioneiro
Otávio Teixeira; Roberto de Oliveira . . . 163
Representación y Clasificación de Datos Geoespaciales Usando Redes Neuronales
Luis Fernando Niño Vasquéz; Marly Esther De Moya Amarís . . . 175

Strength By Objective: Una nueva estrategia de asignación de fitness para Algoritmos Evolulutivos Multi-Objetivos
Guillermo Garcia; Wilmer Pereira; Gonzalo Ron 187
An Architectural Model to Support Collaborative Work through Mobile Devices
César Collazos; Luis Guerrero; Rosa Alarcón 195
Ambiente Computacional para la Edición Cooperativa de Diagramas de Clases con UML
Carlos Cobos; Leydy Muñoz; Jaime Salazar 203
Mecanismos de Flexibilização de Sistemas Gerenciadores de Workflow para Antecipação de Tarefas
Igor Fábio Steinmacher; Fabrício Ricardo Lazilha; Edson Alves de Oliveira Junior; José Valdeni de Lima 215
Patrones de Integración de Servicios Groupware
Federico Naso; Alejandro Fernandez 227
Non-deterministic Regular Positive Negative Inference NRPNI
Gloria Inés Alvarez; Antonio Cano; José Ruiz; Pedro García 239
Un Nuevo Modelo de Detector de Colisiones Basado en Elipsoides
Olmedo Arcila Guzmán; José María Bañón Pinar 251
Método de Visualización de Volúmenes sobre Superficies - VoS
Patricia Shirley Herrera Cateriano; Luis Gustavo Nonato 261
Estendendo o Modelo de Máquina Geométrica a um Ambiente de Programação Visual
Diego Galho Prestes; Renata Hax Sander Reiser; Marcos Borba Cardoso; Antônio Carlos da Rocha Costa 273
Correspondencia de Puntos empleando Algoritmos Genéticos para el Registro de Múltiples Vistas de Imágenes de Rango
John Branch; Sandra Mateus; German Sanchez; Flavio Prieto; Idanis Diaz 285
An MPEG-7 Learning Space for Semantic Image Classification
Andres Dorado; Witold Pedrycz; Ebroul Izquierdo 297
Un algoritmo ACO para la Compresión de Imágenes
Cristian Martinez 309
Software para procesamiento de imágenes
Claudia Andrea Barbosa; Heidy Paola Morales 321
Segmentación vascular y caracterización de placas ateroscleróticas en imágenes de tomografía computarizada 3D
María Alejandra Zuluaga; Luis Felipe Uriza C.; Sergio Iván Mesa; Marcela Hernández Hoyos 333
Uso de técnicas de visualización 3D para la enseñanza de anatomía y fisiología del riñón humano
Oscar Ariza; Gustavo Valbuena; Pablo Figueroa; Santiago Leal; Gabriel Martinez 345
Umbralización Multinivel Usando el Modelo de Color HSI
César Julio Bustacara Medina; Enrique González Guerrero 355
Sistemas Multi-Agente ubícuos para la recuperación de información adaptada
Angela Cristina Carrillo Ramos; Marlène Villanova-Oliver; Jérôme Gensel; Hervé Martin . . 367
A Fault-Tolerant Home-Based Naming Service For Mobile Agents
Camron Tolman; Carlos Varela 379
Diseño de un Soporte Didáctico para Asistentes Personales Digitales que Incorporan la Tecnología Bluetooth
Maria E. Villapol; Mayerling Marquez; Jossie Zambrano 393
Test Rápido de Planificabilidad para R.M. o D.M.
José Manuel Urriza; Javier Dario Orozco; Ricardo Cayssials 405
CRM Shopping Portal for B2C E-commerce based on Relevance Feedback
Giner Alor Hernandez; Lucio Daniel Castelan Vega; Pedro Ariza Acevedo; Jose Oscar Olmedo Aguirre 413
An Evaluation Mechanism for Procedural SQLf
Marlene Goncalves Da Silva; Leonid Tineo 425
SQLf Horizontal Fuzzy Quantified Query Processing
Leonid Tineo 435

Fuzzy Object-Oriented Database Language
Julio Luna; Leonid Tineo 445
Generación de mapas de distribución potencial de especies utilizando minería de datos y sistemas de información geográfica
Manuel F. Vargas Del Valle; Gabriela Marín Raventós 453
Implementación de Primitivas SQL para Reglas de Asociación en una Arquitectura Fuertemente Acoplada
Mario Fernando Guerrero; Camilo Andrés Cerquera; Mario Fernando Diaz; Ricardo Timarán Pereira; Stiven Armero 465
Perspectivas de Conocimientos en Grids de Datos
Luis Cadenas; Emilio Hernández 477
Verificación de Restricciones de Integridad en Transacciones Distribuidas sobre un Cluster de Bases de Datos Relacionales
Stephane Gançarski; Hubert Naacke; Claudia León; Pablo Santini; Martha Rukoz 489
Best Link Contribution: A Multicast Routing Algorithm Class
Reinaldo Vallejos; Alejandra Zapata; Veronica Gacitua 501
HuyaWeb: un motor de búsqueda para imágenes basado en contenido
Robinson Rivas-Suarez; Eduardo Muñoz; Nilka Toro 513
Servidor Genérico Para Portales De Voz (SGVP)
Harold Cruz; Diana Muñoz 525
Security and Integrity Issues Related To Instant Messaging
Xiang Dai; Ernst Leiss 537
Construcción de un cifrador basado en una sola permutación
Hernando Castañeda Marin; Wladimir Rodriguez Graterol 549
A framework for performance evaluation of parallel applications on the Grid
Carlos Figueira; Eduardo Blanco; Emilio Hernández 561
An Electronic Marketplace: Agent-based Coordination Models for online Auctions
Ayse Morali; Carlos Varela; Leonardo Varela 573
Equipo Elitista de Algoritmos Evolutivos Multiobjetivo
José Manuel Fernandez Giangreco; Benjamín Barán 587
Jadima: Arquitectura de Máquina Virtual para la Construcción de Aplicaciones JAVA en Plataformas Grids
Yudith Cardinale; Jesus De Oliveira; Eduardo Blanco 599
Estratégia para desenvolvimento de programas paralelos visando diminuir a intrusão causada por teste de software
Leonardo Albernaz Amaral; Eduardo Augusto Bezerra 611
Reachability: a constrained path propagator implemented as a multi-agent system
Luis Quesada; Yves Deville; Peter Van Roy 623
Developing MAS Solutions with Gaia and AUML
Luca Cernuzzi; Franco Zambonelli 635
Extracción de programas a partir de pruebas: Certificación de la Forma Normal de la Reducción de Polinomios.
Gilberto Pérez; José Luis Freire; José María Molinelli 647
Sistemas de Especificación de Transiciones Probabilísticas: Formato de Reglas y Bisimulación como Congruencia.
Ariel Marcelo Fiuri 659
An algebra for describing features
Pablo Ernesto Martinez Lopez; Jeronimo Irazabal 671
Using Stochastic NTCC to Model Biological Systems
Carlos Olarte; Camilo Rueda 687
Verification of Rewrite Based Specifications using Proof Assistants
Thomas Santana; Mauricio Ayala-Rincon 699
Prueba de propiedades en la caché de un servidor funcional de video bajo demanda
J. Santiago Jorge; Alberto Valderruten; Víctor M. Gulías; David Cabrero 711

Cálculo para el Modelamiento Formal de Arquitecturas de Software Basadas en Componentes y con Restricciones en Tiempo de Ejecución
Henry Alberto Diosa; Juan Francisco Díaz Frías; Carlos Mauricio Gaona Cuevas 723
Semiring-based Fuzzy Constraints in Concurrent Constraint Programming
Alberto Delgado; Jorge Andres Perez; Carlos Olarte; Camilo Rueda 735
Predicción de Pseudonudos en la Estructura Secundaria de ARN via Gramaticas Estocasticas Independientes del Contexto
Marcos Niño; Rafael Garcia 747
Geração Automática de Interfaces para Aplicações Extensíveis por Usuários Finais
Sergio Roberto da Silva; Marco Aurélio Jeronymo 759
Uma Ferramenta Interativa para Visualização e Exploração de Imagens Médicas
Leandro Iglezias; Isabel Manssour; Ricardo Czekster 771
Ulisses: Um Navegador Conceptual para Topic Maps
Giovani Rubert Librelotto; Pedro Rangel Henriques; José Carlos Ramalho 783
Catalogación y búsqueda semántica en un sitio web
Juan Manuel Barrios N.; Claudio Gutierrez 795
BDNG - Una propuesta para el diseño e implementación de Bibliotecas Digitales: BDEAFIT un caso práctico
Edwin Montoya; Maryem Ruiz; Jorge Giraldo; Cástulo Ramirez; Gloria Ospina 809
Experiencias en el Desarrollo de una Plataforma de Servicios de e-Commerce para una Comunidad Virtual de Negocios
Martha Mendoza; Jorge Moreno; Luz Marina Sierra; Roberto Naranjo 821
Power and Real-Time Requirements Integrated with an Adaptive Resource Reservation Soft Real-Time Scheduler
Rodrigo Santos; Javier Orozco; Maria Belen D'Amico; Leonardo Ordínez; Pedro Doñate . . . 833
An LP based algorithm for the Privatized Rural Postman Problem
Julián Aráoz; Oscar Meza; Elena Fernández 845
Grasp en la Resolución del Problema del Clustering
Erick Vicente; David Mauricio; Luis Rivera 857
A Lexical and Syntactical Analyzer for the Exporting of QNAP to the Performance Model Interchange Format (PMIF)
Jerónia Rosselló; Connie U. Smith; Catalina M. Lladó; Ramón Puigjaner 869
An Infrastructure for Implementing Compilers for Concurrent Abstract State Machine Languages
Kristian Magnani; Roberto Bigonha; Mariza Bigonha; Vladimir Iorio; Fabiola Oliveira 881
Evolución de la selección dinámica de métodos en Java
Víctor M. Gulías; Carlos Abalde; David Cabrero; Alberto Valderruten 893
Memory consumption analysis of Java smart cards
Pablo Giambiagi; Gerardo Schneider 905
Un Editor de Programas por Refinamiento
Ascánder Suárez; Vicente Yriarte; Wendi C. Urribarrí 917
XSCoRe: A Program Comprehension Workbench
Pablo Montes; Silvia Takahashi 929
Dos Requisitos À Qualificação de Componentes: uma Abordagem Baseada em Refinamentos Sucessivos
Fernando Vieira Paulovich 943
Ugatze : Model Driven Engineering for Component Reuse
Philippe Aniorte; Seyler Frédérick 955
Gerenciamento de Riscos para Projetos de Software
Pascale Rocha; Arnaldo Belchior 965
Mejoramiento del proceso de pruebas en un contexto de desarrollo de software globalizado
Rubby Casallas; Nicolas Lopez; Dario Correal 977
¿Como estimar la calidad de un Web service?
Maria Pérez; Anna C. Grimán; Luis E. Mendoza 989

Proposta de um Mecanismo de Desenvolvimento e Customização de uma Fábrica de Software Orientada a Domínios
José Augusto Fabri; André Trindade; Roberto Durscki; Marcelo Schneck de Paula Pessôa; Mauro Spínola 1001
A Measurement-based Approach for Improving Software Practices
Liliana Guzmán; Marcello Visconti 1015
Geração de Sistemas de Gestão de Conteúdo com Softwares Livres
Fernando Parreiras; Marcello Bax 1027
Una Experiencia con Estudiantes para la Estimación del Esfuerzo de cada Iteración en Proyectos de Software
José Antonio Pow-Sang Portillo 1037
Validating Dynamic Software Architectures using Alloy
Nazareno Aguirre; Sonia Permigiani; María Marta Novaira 1049
Métrica para la Valoración de las Competencias Humanas en el Proceso Software
Silvia T. Acuña; Ricardo D. Cordero; Saritha G. Figueroa 1059
"LIDIS": Consideraciones para la Generación, Maduración y Aplicación de conocimiento en Ingeniería de Software
Hugo Fernando Arboleda; Jaime Alberto Chavarriaga 1071

Artículos XII Concurso Latinoamericano de Tesis de Maestria **1081**
InteGrade: Middleware para Computação em Grade Oportunista
Andrei Goldchleger; Fabio Kon 1083
Herramienta de Minera de Consultas para el Diseño del Contenido y la Estructura de un Sitio Web
Barbara Poblete; Ricardo Baeza-Yates 1103
Protocolo de Coherencia de Memoria Cache para Sistemas Distribuidos
Jose L. Aguilar Castro; Rodolfo L. Sumoza Matos 1123

Indice de autores / Index of Authors **1136**

Indice de claves / Index of keywors **1139**

Prólogo

La conferencia CLEI2005 ha sido organizada en Cali, Colombia, con el decidido apoyo de varias instituciones. Las dos más prestigiosas Universidades de la ciudad, la Universidad del Valle y la Universidad Javeriana - Cali, se han unido con el Parque Tecnológico de Software de Cali, del sector empresarial, para conformar el grupo interinstitucional organizador del evento.

CLEI2005 es la versión del año 2005 de los encuentros anuales de informática promovidos desde hace ya mas de treinta años por el CLEI, Centro Latinoamericano de Estudios en Informática. Esta organización se ha dado a la tarea de fomentar el desarrollo de la informática en los países latinoamericanos, propiciando el intercambio de avances regionales en investigación y docencia, así como de los desarrollos tecnológicos novedosos de impacto en el sector productivo. La organización CLEI está conformada por 74 universidades e instituciones prestigiosas de Latinoamérica, España y Estados Unidos, y que cubren una amplia gama de intereses científicos y tecnológicos en informática.

La conferencia CLEI2005 agrupa seis eventos:

1. La XXXI conferencia de informática
2. El XIII congreso Iberoamericano de educación superior en computación (CIESC)
3. La 3ra "Latin American Networking conference" (LANC)
4. El XII CLEI/ACM concurso de tesis de maestría en informática
5. El primer congreso colombiano de computación
6. El tecnopolo Colombiano de Software.

Ha sido grato constatar en esta oportunidad la gran dinámica, cimentada por una tradición académica ya establecida, de la investigación latinoamericana en informática.

La calidad y el creciente número de ponencias recibidas de todos los países de la región y de algunos extrarregionales da cuenta de la importancia que ha adquirido la conferencia en el ámbito latinoamericano.

El lector de estos anales podrá constatar la amplitud de intereses y la pertinencia científico-tecnológica de las investigaciones regionales en Informática asi como las reflexiones novedosas sobre la pedagogía de la disciplina en nuestros países.

Organizar un evento de esta magnitud requiere el concurso de muchas instituciones, tanto del mundo académico como del sector productivo. Queremos destacar muy especialmente los patrocinios del Instituto Colombiano de apoyo a la ciencia (Conciencias), de IFIP/ACM en la organización de LANC y de la Universidad Nacional de Medellín.

Nuestros agradecimientos van también al comité directivo del CLEI y a ese gran nùmero de personas, investigadores, miembros de los comités de programa, conferencistas y tutorialistas invitados, panelistas, junto con el grupo entusiasta de estudiantes y personal de apoyo. Su concurso nos ha permitido mantener en esta reunión en tierras Colombianas el prestigio de la más importante congregación de informática a nivel latinoamericano.

Camilo Rueda
Pontificia Universidad Javeriana - Cali
Pte. Comité de Programa

Juan Francisco Díaz
Universidad del Valle
Pte. Comité Organizador

Comité de Programa
XXXI Conferencia Latinoamericana de Informática

Presidente
Camilo Rueda (Pontificia Universidad Javeriana-Cali, Colombia)
Co-presidente
Marta Millán (Universidad del Valle, Cali, Colombia)

Ana Maria Moreno (España)
Maria S. Pérez (España)
Oswaldo Velez-Langs (España)
José Santos Reyes (España)
Julio Cesar Lopez Hernandez (Brasil)
Carlos Agon (Francia)
Jean Besson (Francia)
Andrés Jaramillo (Colombia)
Demetrio Arturo Ovalle (Colombia)
Jesús Antonio Acosta (Venezuela)
Aurora Trinidad Ramirez Pozo (Brasil)
César Muñoz (Estados Unidos)
Olga Mariño (Canada)
Deborah Ribeiro Carvalho (Brasil)
Arturo Hernández Domínguez (Brasil)
Marco César Goldbarg (Brasil)
Elizabeth Goldbarg (Brasil)
Leonardo Matos (Brasil)
Cesar Alberto Collazos (Colombia)
Rodrigo Cardoso (Colombia)
Leila Silva (Brasil)
Enrique González (Colombia)
Álvaro Arenas (Inglaterra)
Ruby Casallas (Colombia)
Alberto Valderruten (España)
Catalina M. Llado (España)
Ramon Fabregat (España)
Francisco Jose Quiles (España)
José Miguel Benedí Ruiz (España)
Ana Pont (España)
Catalina Mostaccio (Argentina)
Nazareno M. Aguirre (Argentina)
Silvia Teresita Acuña (Argentina)
Marcelo Naiouf (Argentina)
Manuel Bermudez (Estados Unidos)
Hugo Banda (Ecuador)
Gustavo Chafla (Ecuador)
Ernesto Cuadros-Vargas (Perú)
Ernst L. Leiss (Estados Unidos)
Ramon Puigjaner (España)
Luis Rivera (Perú)
Nora La Serna (Perú)
Oscar Meza (Venezuela)
Haydemar Nuñez (Venezuela)
Judith Barros (Venezuela)
Wilmer Pereira (Venezuela)
Omar Chioti (Argentina)
Javier Orozco (Argentina)
Regina Motz (Uruguay)
Inés Kereki (Uruguay)
Ernesto Ocampo (Uruguay)
Enrique Vargas (Paraguay)
José Lino Contreras (Chile)
Claudio Meneses (Chile)
Sergio Ochoa (Chile)
Andreas Polymeris (Chile)
Héctor Antillanca (Chile)
Eliana Scheihing (Chile)
Mauricio Solar (Chile)
Mauricio Ayala (Brasil)
Luis Omar Quesada (Bélgica)
Angel García (Colombia)
Juan Francisco Díaz (Colombia)
Andrés Dorado (Colombia)
Ruby Casallas (Colombia)
Lourdes Ortiz(Venezuela)
Isabel besembel (Venezuela)
Giovanny Lucero (Brasil)
Rodrigo Cardoso (Colombia)
Josep-Luis Ferrer (España)
Bartomeu Serra (España)
Pere-Pau Sancho (España)
Francisco Perales (España)
Rodrigo Santos (Argentina)

Revisores Adicionales

Adelaide Bianchini (VE), Adenilso Simao (BR), Alceu Brito (BR), Alejandro Gutierrez (UY), Alvaro Ruibal URUGUAY, Andrea Delgado (UY), Andres Becerra Sandoval (CO), Andres Rodriguez (AR), Andriea Jesus (BR), André Luís dos Santos Domingues (BR), Andrés Adolfo Navarro Newball (CO), Andrés Arcia M (VE), Angel Baigorri (BR), Anita Fernandes (BR), Antal A. Buss (CO), Antonio Fernández-Caballero (ES), Armando De Giusti (AR), Beatriz Eugenia Florián Gaviria (CO), Ben Di Vito (US), Blai Bonet (VE), Boris Mejias (BE), Carlos Alberto Estombelo Montesco (BR), Carlos Humberto Llanos (BR), Carlos Figueira (VE), Carlos Luna (UY), Carlos Noguera (CO), Carlos Olarte (CO), Catalina Fellegi (UY), Cecilia María Lasserre (AR), Celia Ghedini Ralha (BR), Christiane Metzner (VE), Claudia León (VE), Claudia Russo (AR), Claudio Delrieux (AR), Cláudia Nalon (BR), Concettina Di Vasta Valente (VE), Cristina Gonzalez (CO), Danilo Bassi (CL), Dario Ernesto Correal (CO), Diego Ernesto Leal (CO), Diego Ordoñez (BE), Dionisio Acosta (VE), Domingo Sebastian (ES), Dulce M Rivero A (VE), Edmundo Leiva (CL), Edna Ruckhaus (VE), Eduardo Fernandez (UY), Eloina Mesa (CO), Emely Arraiz (VE), Emilio Ormeño (AR), Enrique Arias (ES), Enrique Latorres (UY), Erick Araya (CL), Esteban Barrios (UY), Fabio Araujo (BR), Fernando Carpani (UY), Fernando De la Rosa (CO), Fernando Machado (UY), Flor Nacriso F (VE), Flor Narciso F (VE), FlÃ¡vio L. C. de Moura (BR), Francisco Jose Perales (ES), Francisco Moreno (CO), Fred Spiessens (BE), Gabriel Gambetta (UY), Gabriel Tamura (CO), Gareth Barrera (CO), Gerardo Matturro (UY), Gilles Dowek (FR), Giulliana Souza (BR), Graciela Ferreira (UY), Guillermo Calderón (UY), Gustavo Ospina (BE), Harold Castro (CO), Haydemar Núñez (VE), Hugo Ramán (AR), Hugo Vecino (CO), Héctor Soza (CL), Isabel Besembel (VE), Ivan Santin (BR), Jacqueline Zaccara (UY), Jaime Alberto Parra (CO), Jaime Guzmán (CO), Jean Iratchet (CL), John William Branch (CO), Jonás A. Montilva C. (VE), Jorge Carlos Lucero (BR), Jorge Henrique Cabral Fernandes (BR), Jorge Salas (VE), Jorge Triñanes (UY), Jose Pascual Molina (ES), Josep-Lluis Ferrer-Gomila (ES), José Miguel Puerta (ES), José Pascual Molina Massó ((ES)), José Alvarez G. (CL), Juan Carlos Garcia Ojeda (CO), Juan David Velásquez (CO), Juan Moreno (UY), Judith Barrios A (VE), Laura Lanzarini (AR), Laura Sanchez Garcia (BR), Luca Cernuzzi (PY), Lucia Cardoso (VE), Luis Mariano Bibbo (AR), Luis Lobos (CL), Luis Orozco (ES), Luiz Oliveira (BR), Mabel del V. Sosa de Goldar (AR), Mabel del Valle Sosa (AR), Magali Gonzalez (PY), Marcela Varas (CL), Marcelo Arroyo (AR), Maria Gertrudis López (VE), Maria Istela Cagnin (BR), Maria T. López (ES), Maria-Isabel Sanchez-Segura (ES), Mariel Feder (UY), Marta Lopez Fernandez (ES), Marta Millan (CO), Martin Musicante (BR), Martin Solari (UY), María E. Urquhart (UY), María Gertrudis López (VE), María Rosa Galli (AR), Mauricio Ayala-Rincon (BR), Mauricio Sepúlveda (CL), Miguel Castro (VE), Miguel Torrealba (VE), Miguel Torres (CO), Mónica Wodzislawski (UY), Nicolás C.A. Antezana Abarca (PE), Nicolás López (CO), Nora Montaño (VE), Nora Szasz (UY), Oduvaldo Bessa (BR), Oscar Bedoya (CO), Oscar Bria (AR), Oscar Dieste (US), Pablo David Villarreal (AR), Pablo Figueroa (CO), Pascual Gonzalez (ES), Patricia Pesado (AR), Patrick O'Callaghan (VE), Pedro Pinacho CHILE, Pedro Salvetto (UY), Pierre Dupont (BE), Radu Siminiceanu (US), Rafael Novales (UY), Raimundo Vega (CL), Renaud De Landtsheer (BE), Rhadamés Carmona (VE), Ricardo Timarán (CO), Roberto Alves (BR), Robinson Rivas-Suarez (VE), Rodolfo Bertone (AR), Rosa Muñoz (CL), Rubén Recabarren (VE), Saritha G. Figueroa (AR), Sebastian Gonzalez (BE), Sergio Zapata (AR), Silvia Takahashi (CO), Silvia Vergilio (BR), Silvio Gonnet (AR), Sira Vegas (ES), Stefano Gualandi (IT), Valentin Mesaros (BE), Valter Camargo (BR), Vianca Vega (CL), Vicente Gonzalez (PY), Vicente González (PY), Vicente Ramirez (VE), Victor Carreno (US), Viviana Elizabet Quincoces (AR), Víctor M. López Jaquero (ES), Waldo Cancino (BR), Washington Salaberry (UY), William Torrealba (VE), Wladimir Rodriguez (VE), Yussef Farran (CL)

Comité de Programa
XII Concurso Latinoamericano de Tesis de Maestría

Jurados

Cristina Cornes
Inés Kereki
Nora Szasz
Álvaro Tasistro

Evaluadores

Ines Friss de Kereki (UY)
Alberto Restrepo (CO)
Hernan Astudillo (CL)
Horst von Brand (CL)
Juan Lalinde (CO)
Isabel Besembel (VE)
Ernst Leiss (US)
Claudio Meneses (CL)
Carlos Castro (CL)
Silvio Gonnet (AR)
Pablo David Villarreal (AR)
Maria Laura Caliusco (AR)
Itana Gimenes (BR)
Carlos Neil (AR)
Gabriela Arevalo (AR)
Claudia León (VE)
Pablo E. Martínez López (AR)
Sandra Fabbri (BR)
Roberto Bigonha (BR)
Alvaro Tasistro (UY)
Gustavo Betarte (UY)
Carlos Luna (UY)
Maria Cristina Riff (CL)
Pere Palmer (ES)
Diego Andrade (EC)
Raimundo Vega (CL)
Margaret Miró-Julià (ES)
Solange Oliveira Rezende (BR)
Hector Cancela (UY)
Fernando Paganini (UY)

Comité Organizador
CLEI2005

Comité Directivo
Juan Francisco Díaz
Camilo Rueda
Orlando Rincón

Comité de Actividades Culturales y atenciones especiales
Maria Eugenia Valencia
Bertha Bolaños

Comité de Difusión y Enlace
Luis astorquisa
Olga Lucía Gálvez
Jessica Rodriguez
Angela Villota
Luis Alejandro Giraldo
Jhon Iván Mondragón
Rommy Bitar
Oscar Fernando López

Comité de Servicios de Información
Mauricio Gaona
Antal A. Buss
Mario Alberto Cruz
Carlos Muñoz

Comité de Logística
Olga Lucía Delgadillo
Elicenia Castrillón
Ivette Ortiz
Nubia Aponte
Andrés Becerra
Gerardo Sarria
Claudia Timaná

Comité Show Room
Diana Holguin
Diana Dávila

Asistencia Operativa
María Cecilia Villegas
Ana María Rodríguez

Coordinación con comités académicos
Angel García
Ramón Puigjaner
Alvaro Tasistro
Inés Kereki

Artículos XXXI Conferencia Latinoamericana de Informática

A New Similarity Measure for Comparing Induced Binary Decision Trees

G. Müller B.
Universidad Central, Facultad de Ingeniería,
Caracas 1040A, Venezuela, Apartado 47937
gmullerb@mail.com

and

C. M. Rocco S.

Universidad Central, Facultad de Ingeniería,
Caracas 1040A, Venezuela, Apartado 47937
crocco@reacciun.ve

Abstract

In this paper a new similarity measure to compare binary decision trees is proposed. The new measure takes into account the structure of the trees and their predictive power. The main differences with respect to other proximity measures proposed in the literature are: First, its calculation does not depend on a given data set so no classification procedure is required; and second the new measure can be used to compare simultaneously different trees. The example analyzed, referred to an electric power system shows the use of the proposed measure.

Keywords: Decision Trees, System Reliability Assessment, Similarity Measures.

Resumen

Este artículo presenta una nueva medida de similitud para comparar árboles de decisión binarios. La medida propuesta considera tanto la estructura del árbol como su capacidad de pronóstico. Las diferencias más importantes con respecto a otras medidas definidas en la literatura son: 1) el cálculo no depende de un conjunto de datos específico, por lo que no requiere una fase de clasificación; y 2) la medida puede ser utilizada para comparar más de dos árboles simultáneamente. El uso de la medida propuesta se ilustra con un ejemplo relacionado con la evaluación de la fiabilidad de un sistema eléctrico de potencia.

Palabras claves: Arboles de Decisión, Evaluación de la Fiabilidad en Sistemas, Medidas de Similitud.

1. INTRODUCTION

Decision Tree (DT) based methods are non-parametric classification approaches, that have been used as a machine learning technique useful in the analysis of large data sets for which complex data structures may exist [1]. Like any classification method, the task of a DT is to obtain a model of the system at hand, starting from a finite collection of examples generated according to an unknown or complex function describing the behavior of the system. The structure of the induced DT not only defines the most important attributes of the system considered but also the rules that could be extracted.

Binary DTs have been applied in [2] to speed up a Monte Carlo simulation for system reliability assessment and to obtain approximate reliability expressions by means of the extraction of rules from the induced tree [3]. The main idea was to develop an estimation algorithm based on a DT by training a model on a restricted random data set. Then the induced tree replaces the System Function (SF), which determines if a given system state is of success or failure by a simpler calculation, which provides reasonably accurate model outputs.

One of the most commonly referred problems in DT applications is the instability of the induced tree, that is, different training data sets do not necessarily produce the same tree structure even if their predictive power could be similar.

This characteristic is of utmost importance in the approaches proposed to assess the reliability of a system [2,3], since the training set is randomly generated and the rules that approximate the system behavior must be extracted from just one DT.
Another area in which the comparison among trees is also important is the one related to the extraction of comprehensible models from black-box models, like artificial neural networks (ANN) or support vector machines (SVM). Indeed, one common approach is to use a specific algorithm to extract a decision tree from the black-box model and then convert it into a set of IF-THEN rules. For example, the well-known TREPAN algorithm has been successfully used to extract decision trees form ANN [5] and, recently, from SVM [6]. But extracted trees are only evaluated through performance indexes like sensitivity, specificity and accuracy [7] without considering the structure of the extracted trees, which determines their equivalent set of rules.
To cope with this type of uncertainty, two approaches have been suggested. The first consists on combining different DTs, (e.g. bagging and boosting) which has proved to reduce the variance of the equivalent classifier, even if the combination can't be interpreted as an equivalent tree structure. The second approach analyzes a set of DTs in search of a single model that can explain different aspect of the data [4].
In order to compare or to evaluate the similarity between different induced trees different proximity measures have been defined. These measures evaluate the difference between the structures of two trees, compare their predictive power or take into account both aspects. An excellent review of the existing measures can be found in [4].
This paper proposes a new measure, which evaluates the similarity between binary DTs considering their associated partitions and predictions as well as their structures. Since the proposed measure does not use any specific data set to evaluate the similarity among induced trees, its calculation is faster than the existing similarity measures since no classification procedure is required. Additionally, the proposed similarity measure can be used to compare several trees simultaneously.
To evaluate the proposed proximity measure, a comparison between binary DTs generated using different random training sets related to an electric system reliability assessment is presented. The comparison also considers the proximity measures suggested in [4].
The organization of the paper is as follows. Section 2 present the existing measures to compare different decision trees. The proposed measure is described in Section 3. Section 4 presents the results of the comparison of several DT induced using a Power Network example. Finally, Section 5 presents the conclusions.

2. CRITERIA TO COMPARE DECISION TREES

This section describes some techniques previously developed to compare different decision trees. From two data sets D_i and D_j, two binary decision trees T_i and T_j are induced. Each tree offers its description of the data. In order to compare T_i and T_j three criteria are possible. The first criterion is based on the observed response and/or the partition defined by terminal nodes [4]. The second criterion is based on the comparison of the tree structure. Finally, T_i and T_j can be compared using a hybrid criterion, which takes into account the structure and the predictive power.

2.1 Proximity measures based on prediction and partition

To compare two decision trees T_i and T_j, a data set D with r observation (D={D(1), D(2), ...D(r)}) not used in D_i and D_j, is selected. The following variable is defined [8]:

$$w_k = \begin{cases} 1 & \text{if } T_i(D(k)) = T_j(D(k)) \\ 0 & \text{otherwise} \end{cases} \qquad k=1,..,r$$

where $T_i(D(k))$ is the classification produced by tree T_i, when it evaluates the k-th observation of D.
Then a similarity index for T_i and T_j is:

$$Sim(T_i,T_j) = \left(\sum_{k=1}^{r} w_k \right) / r$$

This index is 1 if all of the r observations are classified in the same way by decision tree T_i and T_j, and 0 if all of the classifications are not equal.
Miglio and Soffritti [4] propose a similarity index based on the classification rules that the trees represent. Let T_i and T_j be two different trees with H and K leaves respectively that can be used to classify a set of r observations. Leaves (or terminal nodes) of T_i, are labeled from one to H, and leaves of T_j from one to K. The matrix $\mathbf{M}=[m_{h,k}, h=1,..,H;k=1,..,K]$ defines the number of objects which belong simultaneously to the h-th leaf T_i, and to the k-th leaf of T_j. Then the similarity index $B(T_i, T_j)$ of Fowlkes and Mallows is defined as [4]:

$$B(T_i, T_j) = \frac{U_{ij}}{\sqrt{P_i Q_j}}$$

where:

$$U_{ij} = \sum_{h=1}^{H}\sum_{k=1}^{K} m_{h,k}^2 - n$$

$$P_i = \sum_{h=1}^{H} m_{h,0}^2$$

$$Q_j = \sum_{k=1}^{K} m_{0,k}^2$$

$$m_{h,0} = \sum_{k=1}^{K} m_{h,k} \quad m_{0,k} = \sum_{h=1}^{H} m_{h,k} \quad n = \sum_{h=1}^{H}\sum_{k=1}^{K} m_{h,k}$$

In order to compare T_i and T_j from the point of view of their associated partitions and predictions, Miglio and Soffritti [4] modify the measure $B(T_i, T_j)$, taking into account whether the elements of the partition derived by different trees, assign the same class label to each object.
Let $c_{h,k}$=1 if the h-th leaf of T_i and the k-th leaf of T_j have the same class label and $c_{h,k}$=0 otherwise. Then the similarity between T_i and T_j is defined as [4]:

$$B(T_i, T_j) = \frac{\sum_{h=1}^{H}\sum_{k=1}^{K} m_{h,k}^2 c_{h,k} - \sum_{h=1}^{H}\sum_{k=1}^{K} m_{h,k} c_{h,k}}{\sqrt{P_i Q_j}}$$

2.2 Proximity measures based on topology

In [4] the authors describe a proximity distance, due to Shannon and Banks, which measures "the amount of rearrangement needed to change one of the trees so that they have an identical structure":

$$d(T_i,T_j) = \sum_r \alpha_r \left| S_{ijr} \right|$$

where S_{ijr} denotes the set composed by discrepant paths of length r found on only one of the two trees, and α_r is a weight function that depend only on r. Since this measure is based on the number of paths in T_i and T_j, a normalizing factor (NF) is defined such that $0 \le NF*d(T_i,T_j) \le 1$[4]. This measure "can be interpreted as the number of discrepant paths out of the total number of paths present in the pair of trees" [4].

2.3 Proximity measures based on structure and the predictive power

In order to compare two decision trees, Miglio and Soffritti [4] propose a dissimilarity measure which takes into account the fact that "trees having the same distance with respect to their structure can show different predictive powers, and that trees with the same predictive power can have different structure". The proposed measure is defined as:

$$\Delta(T_i, T_j) = \frac{\delta(T_i, T_j)}{\max \delta(T_i, T_j)}$$

where:

$$\max \delta(T_i, T_j) = \sum_{h=1}^{H} \alpha_h \frac{m_{h,0}}{n} + \sum_{k=1}^{K} \alpha_k \frac{m_{0,k}}{n}$$

$$\delta(T_i, T_j) = \sum_{h=1}^{H} \alpha_h (1 - s_h) \frac{m_{h,0}}{n} + \sum_{k=1}^{K} \alpha_k (1 - s_k) \frac{m_{0,k}}{n}$$

$s_h = \max(s_{h,k}, k=1,..,K)$

$$s_{h,k} = \frac{m_{h,k} c_{h,k}}{\sqrt{m_{h,0} m_{0,k}}}, \; k=1,..,K$$

Since $\Delta(T_i,T_j)$ is a dissimilarity index, $\Delta(T_i,T_j)$ is 0 when the trees T_i and T_j are equal.

3. THE PROPOSED MEASURE

The measures presented in 2.1 and 2.3 are based on the use of a given data set D to calculate the similarity between two DTs. The proposed similarity measure $\Theta(T_i,T_j,..,T_L)$ does not require any classification procedure and can be used to compare simultaneously two or more trees, taking into account their structures and the classes of the terminal nodes.
Every node in a DT is associated with a binary attribute. For each node there are two exit branches. Every terminal node (or leaf node) is associated with a binary class.
For example, consider DTi shown in figure 1. It has four decision nodes y =1,..,4, and five leaf nodes r_v, v=1,..,5, each one representing a classification class (C_1 and C_2).
The matrix $\mathbf{N} = [n_{x,y}, x = 1,.., L; \; y = 1, ..., \max(\mathbf{NS}_x)]$ contains all the information regarding the nodes of all trees, where $\mathbf{NS}_x$, x=1,..L, is the number of nodes in T_x, and L is the number of trees to be compared.
The function $GN(n_{i,h},n_{j,k})$, which compares the similarity between two nodes (h and k) in two different trees T_i and T_j as: $0 \le GN(n_{i,h},n_{j,k}) \le 1$; ($h \in T_i$ and $k \in T_j$) is defined as:

$$GN(n_{i,h},n_{j,k}) = \begin{cases} 1, \text{ if node } h \in T_i \text{ and node } k \in T_j \text{ are equal, and the branches leaving both nodes are equal and the terminal nodes are equal.} \\ 0.5, \text{ if node } h \in T_i \text{ and node } k \in T_j \text{ are equal, and only one branch leaving both nodes is equal.} \\ 0, \text{ otherwise.} \end{cases}$$

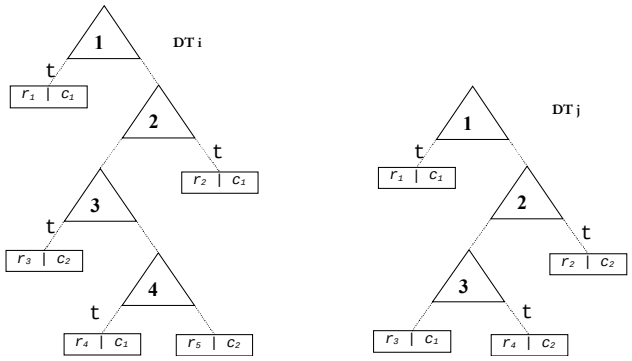


Fig. 1. Two DTs to be compared.

For example, let's consider DTs T_i and T_j in figure 1. Then:

$GN(n_{i,1},n_{j,1}) = 1$
$GN(n_{i,2},n_{j,2}) = 0.5$
$GN(n_{i,4},n_{j,3}) = 0$

That means that the proposed measure takes into account the splitting variable and the value it splits on. To compare T_i and T_j it is necessary to compare each node $h \in T_j$ against all the nodes of T_j. During the comparison three situations are possible:

- Node h is not present in T_j
- Node h is present in T_j, with some degree of similarity
- Node h is present several times on T_j, with different degrees of similarity

In order to consider these situations the following values are defined:

The number NI of nodes that are equal to node h in T_i:

$$NI = \sum_{y=1}^{NS_i} GN(n_{i,h}, n_{j,y})$$

The number NE of nodes that are equal to node h in T_j:

$$NE = \sum_{y=1}^{NS_j} GN(n_{i,h}, n_{j,y})$$

Then the function $GNE(n_{i,h},j)$, which compares the degree of presence of node $h \in T_i$ in T_j, is defined as:

$$GNE(n_{i,h}, j) = \begin{cases} \dfrac{NE}{NI} & \text{if } NI > NE \\ \dfrac{NI}{NE} & \text{if } NE > NI \end{cases}$$

From here, the new proximity measure is the defined as:

$$\Theta(Ti,Tj,..,T_L) = \frac{\sum_{x=0}^{L}\left\{\frac{1}{NS_x}\cdot\sum_{y=0}^{NS_x}\left\{\frac{1}{L-1}\cdot\sum_{z=0,z\neq x}^{L} GNE(n_{x,y},z)\right\}\right\}}{L}$$

The expression $\sum_{z=0,z\neq x}^{L} GNE(n_{x,y},z)$ evaluates the presence of node $y \in T_x$ in all the DTs. The expression $\sum_{y=0}^{NS_x}\left\{\frac{1}{L-1}\cdot\sum_{z=0,z\neq x}^{L} GNE(n_{x,y},z)\right\}$ represents the comparison of all the nodes $\in T_x$ against all the DTs to be compared.

Finally $\Theta(T_i,T_j,..,T_L)$ represents the similarity among all the DTs.
The proximity measure can be further modified to include a weighting scheme, which can be used to take into account the level where a specific node in a tree T_i is found in another tree T_j [8].

4. EXAMPLE

Figure 2 shows the electric power network (IEEE 14 bus system [9]) to be used for comparing several induced binary DTs. It is assumed that only transmission lines have two states (failure and success) (failure events are assumed as independent events). The state of the i-th transmission line (x_i) is defined as:

$$x_i = \begin{cases} 1 & (\text{succes state}) \text{ with Probability } R_i \\ 0 & (\text{failure state}) \text{ with Probability } 1 - R_i \end{cases}$$

The j-th state of the system containing NL transmission lines is expressed by the vector $\boldsymbol{x}_j = (x_1, x_2, \ldots, x_{NL})$. To establish if $\boldsymbol{x}_j$ is an operating or a failed state, each power system state is evaluated through a special SF, denoted as DC-load flow (LF):

$$y_j = LF(\boldsymbol{x}_j) = \begin{cases} 1 & \text{if the system is operating in this state} \\ 0 & \text{if the system is failed in this state} \end{cases}$$

That means that the function $LF(\boldsymbol{x}_j)$ is able to define if a given system state $\boldsymbol{x}_j$ correspond to an operating or failed state. To compare different induced trees, a data set with 20000 different pairs $(\boldsymbol{x}_i, y_i)$ is randomly generated. The data set is then divided in ten different data sets, each one with 2000 pairs. Each data set is used to train a DT, which is then tested on an additional data set with 10000 random pairs $(\boldsymbol{x}_i, y_i)$.
In the Monte Carlo approach presented in [2,3], the assessment of the system reliability is performed as follows:

1) A large number N of system states is sampled;
2) Each sample is evaluated through a System Function (SF) to evaluate if the state is of success or failure;
3) A system reliability point estimate is obtained as: $\hat{R} = \dfrac{\text{Total number of Success States}}{N}$

Once a DT is induced, it replaces the SF. In this way we can evaluate from a reliability point of view, how good is the approximation between the SF and the DT.

The following results were obtained for the ten induced trees:

Number of nodes: Range = [117,141], average 135
Training accuracy: Range = [99.40,99.55] %, average 99.464 %
Testing accuracy: Range = [96.62,98.10] %, average 97.375 %
System reliability relative error: [-0.24, -0.17] %, average -0.20 %.

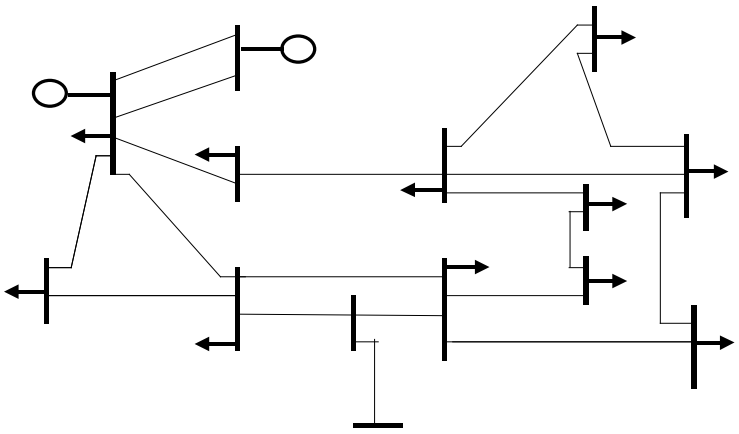


Fig. 2. Single-Line Diagram of IEEE 14 Bus Power Network [6]

From the accuracy point of view, it is interesting to note that the high figure obtained during the testing phase could mean that the induced trees are very similar. As suggested in [4], "this fact is probably due to the presence of predictors, which have a similar discriminant power" but the induced trees can be very distant structurally. It is also interesting to note that the system reliability average error is less than 0.3 %. This means that the reliability assessment using different DTs is not seriously affected by different training sets.
However, in order to evaluate the difference among the induced trees, the similarity measures previously described along with the proposed one were calculated. Table 1 shows the results obtained for the different measures. The values reported correspond to the average similarity between T_i and T_j, i=1,..,10; j=1,..,10; i≠j.
The results show that the induced trees are indeed very similar, not only from a predictive point of view, but also from a structural point of view, as suggested by the similarity measure $1-\Delta(T_i,T_j)$. The proposed similarity measure $\Theta(T_i,T_j)$ suggests the same conclusion.

Table 1. Similarity measures comparison.

Measure	Range	Average	Standard deviation
$Sim(T_i,T_j)$	0.9944-0.9980	0.9961	0.0011
$B(T_i,T_j)$	0.9521-0.9999	0.9672	0.0155
$1-\Delta(T_i,T_j)$	0.9154-0.9936	0.9434	0.0258
$\Theta(T_i,T_j)$	0.9727-0.9991	0.9865	0.0123

It is interesting to note that the average values of the similarity measures are greater than 0.94. That means that, for the system under analysis, the predictive power and/or structure of the induced trees are not seriously affected by different training sets, so any induced tree can be used as a simpler model to replace the SF for reliability assessment or for rules extraction.
The simultaneous comparison of the ten induced trees, evaluated through the proposed similarity measure results in $\Theta(T_1,T_2,..,T_{10}) = 0.9524$. This value also suggests that all of the induced trees are very similar, in concordance with the previous conclusion. However, the evaluation of $\Theta(T_1,T_2,..,T_{10})$ is performed simultaneously.

5. CONCLUSION

In this paper different existing measures to evaluate the similarity among induced binary decision trees were compared with a new proposed measure. The main characteristic of the new measure is that no classification procedure on a specific data set is required and it allows the comparison of several induced trees simultaneously. The results obtained show that the proposed measure seems to be a promising approach for assessing the similarity among different induced binary DTs.
Even if the proposed measure has been defined to evaluate binary induced DT, it can be easily extended to consider DTs composed by multi-valued attributes, for example, when considering multi-state system reliability [10].
The proposed measure could be used to evaluate the similarity among different sets of trees and to determine which set has the most similar trees or to generate a relatively small heterogeneous set of trees, which could improve bagging.
Finally, the use of the proposed measure for identifying a "typical" tree, that is a single tree that could represent all the trees belonging to a set of different trees, is another area to be evaluated.

References

[1] Cappelli C., Mola F., Siciliano R.: A statistical approach to growing a reliable honest tree, *Computational Statistics & Data Analysis*, 38, 2002, pp 285-299
[2] Rocco C.M.: A Rule Induction Approach to Improve Monte Carlo System Reliability Assessment", *Reliability Engineering and System Safety*, 2003, 82/1 pp. 87-94
[3] Rocco S. C. M., Muselli M.: Empirical Models Based On Machine Learning Techniques for Determining Approximate Reliability Expression, *Reliability Engineering and System Safety*,2003, 83/3 pp. 301-309
[4] Miglio R., Soffritti G.: The comparison between classification tress through proximity measures, *Computational Statistics & Data Analysis* 45, (2004), 577 – 593
[5] Craven M.W. (1996): Extracting Comprehensible Models from Trained Neural Networks. Ph.D. Thesis, University of Wisconsin- Madison.

[6] Torres D., Rocco C.M.: A Comparative Study for Assessing the Reliability of Complex Networks using rules extracted from different Machine Learning Approaches", submitted to The 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia
[7] Veropoulos K, Camppbell, Cristianini N.: Controlling the Sensitivity of Support Vector Machines. Proceedings of the IJCAI 1999
[8] Müller G.: Comparing Decision Trees, Universidad Central de Venezuela, Fac. Ingenieria, Internal Report DIOC-2004-04, 2005 (In Spanish)
[9] Billinton, R., Hossain K.L.: Reliability equivalents: Power system applications, *Reliability Engineering,* vol. 5, 1983, 99 239-257
[10] Lisnianski A., Levitin G.: *Multi-State System Reliability*, World Scientific, New Jersey, 2003

Invariant Pattern Recognition by Projection Histograms and Fuzzy ART Neural Network

Guillermo Cámara Chávez, Arnaldo de Albuquerque Araújo
Computer Science Department
Federal University of Minas Gerais
Av. Antônio Carlos 6627 - Campus Pampulha
Belo Horizonte - Minas Gerais - Brazil
gcamarac,arnaldo@dcc.ufmg.br

Abstract

Pattern recognition has provoked a great interest in the last decades due to the use of the computers. The complexity of a pattern recognition system is high because much of the available information in real life is presented in the form of complex patterns, suffering linear transformations and even nonlinear deformations. The objective of this paper is to develop a model for invariant pattern recognition by combining Projection histograms and Fuzzy ART neural networks. The former works as invariant feature extractor while the latter acts as a robust classifier. The considered model will efficiently recognize patterns without taking in consideration the possible variations of position, scale and rotation. The experimental results with our databases shows that the model can achieve high classification accuracy.

Keywords: Invariant pattern recognition, projection histograms, Fuzzy ART.

Resumen

El reconocimiento de patrones ha provocado un gran interés en las últimas décadas debido al uso de las computadoras. Los patrones poseen una alta complejidad porque la mayoría de la información disponible en la vida real esta presente de forma de patrones complejos, sufriendo transformaciones lineales, así como deformaciones no lineales. El objetivo de este artículo es desenvolver un modelo para reconocimiento de patrones invariantes, combinando proyecciones histográmicas y la red neuronal Fuzzy ART. El primero trabaja como extractor de características, mientras que el segundo actúa como un clasificador robusto. El presente modelo reconoce eficientemente patrones sin llevar en consideración posibles transformaciones en posición, escala y translación. Los resultados experimentales con nuestras bases de datos mostró que el modelo es capaz de alcanzar un elevado índice de clasificación.

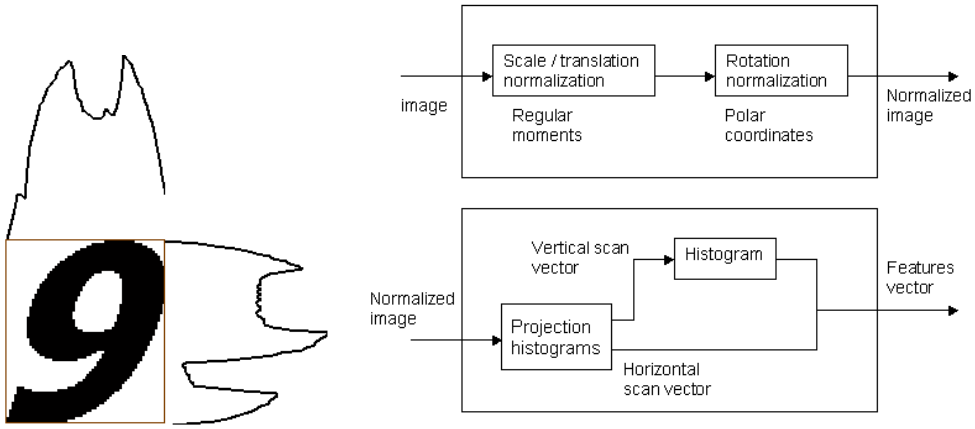
Palabras claves: Reconocimiento de padrones invariantes, proyecciones histográmicas, Fuzzy ART.

1 Introduction

Pattern recognition is the study of how machines can observe the environment, learn to distinguish patterns of interest from their background, and take reasonable decisions about the categories of the patterns [1]. The use in the last decades of computers and electronic devices impulses the study of pattern recognition techniques. Interest in the area of pattern recognition has been renewed due to emerging applications which are not only challenging but also computationally more demanding. Patterns in real life are usually presented in different positions, having variations in rotation, scale and/or translation. So the research of invariant pattern recognition is very important for real applications.

A pattern recognition system usually comprises three main components, namely preprocessing, feature extraction and classification. In the preprocessing stage the input image suffers from a variety of operations such as noise removal, segmentation and image enhancement. Feature extraction aims to represent the image in terms of some quantifiable measurements that may be easily utilized in the classification stage. There are many feature extractors like Fourier descriptors [2], invariant moments [5], wavelets [9]. In the classification stage the patterns are grouped according to similar characteristics. Some of the classifiers are: neural networks, k-means algorithm, etc.

Artificial neural networks (ANN) have proven to be powerful classifiers due to the characteristics of learning, fault tolerance and robustness [4]. ART (Artificial Resonance Theory) was developed by Carpenter and Grossberg [6]. This



(a) (b)

Figure 1: (a) The vertical and horizontal projections of number 9, (b) proposed model.

kind of neural network can learn new patterns without forgetting old knowledge. Specifically, the Fuzzy ART [7] has the capability to learn recognition categories rapidly, in response to arbitrary sequences of analog or binary input patterns.

In this paper we present a model for invariant pattern recognition by combining Projection histograms and Fuzzy ART neural networks. The former works as invariant feature extractor while the latter acts as a robust classifier. The considered model will efficiently recognize patterns without taking in consideration the possible variations of position, rotation and scale. The experimental results show that the model can achieve high classification accuracy.

The organization of this paper is as follows. Section II briefly reviews the Projection histograms. In Section III, we discuss the use of Fuzzy ART neural network as a classifier. Experimental results are presented in Section IV. Section V concludes the paper.

2 Projection Histograms

Projection histograms were introduced in 1956 by Glauberman [3]. Projection histograms counts the number of pixels in each row and column of a binary image (Figure 1(a)), i.e., we will have two vectors, one for vertical projection and another for horizontal projection. These projection histograms could be extended for gray level images.

The proposed model is invariant to rotation, scale and translation, see Figure 1(b). The first step is to normalize an image in translation and scale, for that purpose we use regular moments, then we use polar coordinates for rotation normalization. Now we have a normalized image, and finally we find the projection histograms (vertical and horizontal scan vector). Unfortunately, vertical scan vector is not invariant to rotation, to achieve invariance we find the histogram of the vertical scan vector. Now, we have two feature vectors that are invariant to translation, scale and rotation. We describe our proposed model with more details in the next paragraphs.

Firstly, we need to normalize in translation and scale. To achieve scale and translation uniformity, the regular moments [5] (i.e. m_{pq}) of each image are used. Translation invariance is achieved by transforming an image into a new one whose first order moments, m_{01} and m_{10}, are both equal to zero. This transformation is done by transforming the original $f(x, y)$ image into another $f(x + \overline{x}, y + \overline{y})$, where $\overline{x}$ and $\overline{y}$ are the centroid localization of the original image.

First order moments are used to compute the image's centroid (x_c, y_c):

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}} \tag{1}$$

where m_{00} represents the image's area, m_{10} and m_{01} are projections on x and y, respectively.

Scale invariance is accomplished by enlarging or reducing each shape such that its zeroth order moment m_{00} is set equal to a predetermined value β. The scale invariance is achieved by transforming the original image $f(x, y)$ into a new function $f(x/a, y/a)$ with $a = \sqrt{\beta/m_{00}}$ [8].

Rotation invariance is achieved by transforming the image function $f(x, y)$ into a function $f(\rho, \theta)$ in polar coordinates, where ρ is the length of vector from origin (centroid) to (x, y) pixel and θ the angle between vector ρ and x axis in counterclockwise direction. The rotations in polar coordinates are circular shifts (see Figure 2).

We find the projection histograms after the image is normalized (translation, scale and rotation), and as a result we have two vectors, "horizontal scan vector" and "vertical scan vector". The horizontal scan vectors of the same image with different rotation will be similar due to polar coordinates properties. There exists a significant difference in vertical scan vector, see figure 3.

We can use horizontal scan vector as feature characteristics. The second vector could be used as a discriminator between similar images. That's why we need a way to represent the vertical scan vector. A histogram of that vector supplies a new vector that is rotation invariant, see Figure 4.

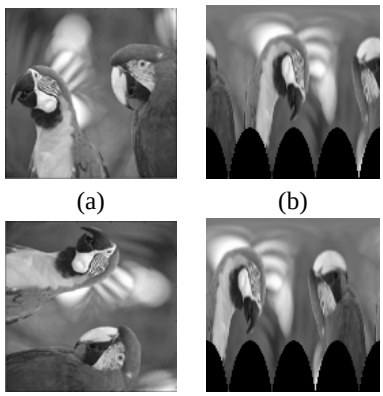


Figure 2: Polar representation of "parrots" image with different rotation angles. The first column shows images in Cartesian coordinates and the second column in polar coordinates.

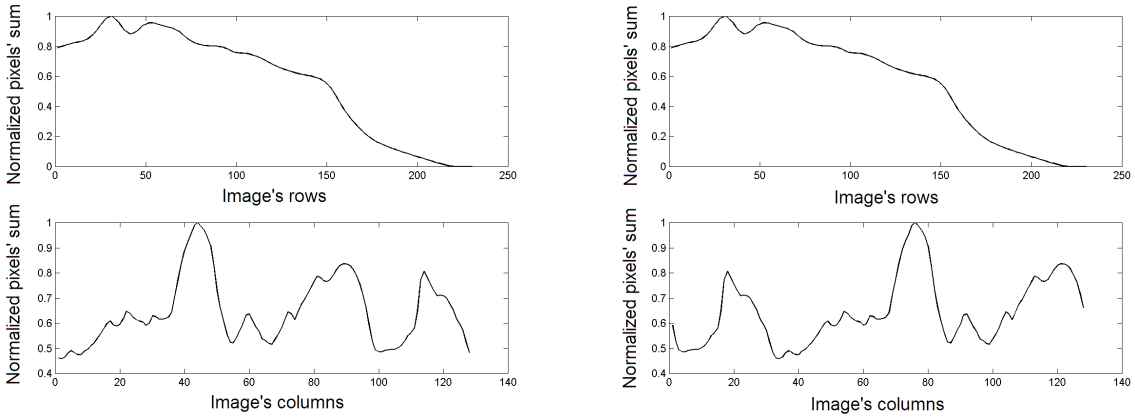


Figure 3: Scan vectors of "parrots" image, (a) horizontal and vertical scan vectors of the image with no rotation and (b) with 90 degrees rotation.

3 Neural network *Fuzzy* ART (Adaptive Resonance Theory)

In 1976, Grossberg introduced the adaptive resonance theory, concept as a cognitive human information process, through a self-organization, effectuating stable category recognition and answering an arbitrary sequence of input patterns [10]. Fuzzy ART was developed by Carpenter and Grossberg [7], this neural network inherits the design

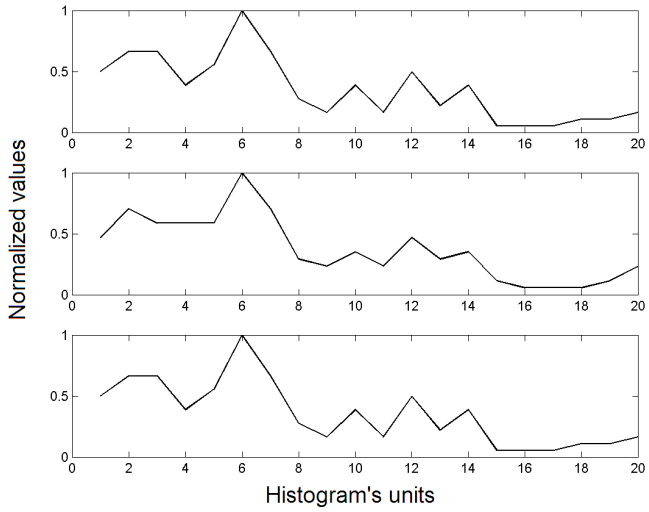


Figure 4: Vertical scan vector histogram of "parrots" image with rotations of $0°$, $45°$ and $90°$ degrees.

features of other ART models, and incorporates computations from fuzzy set theory into the ART1 [6] neural network. As a consequence Fuzzy ART can learn and classify analog patterns. This kind of network has a self-organization and a self-stabilization that allows solving the stabilityplasticity dilemma. Thus, the ART network is capable to assimilate new things while maintaining those already instructed [10].

3.1 Fuzzy ART Algorithm

Each input pattern I is a m-dimensional vector $(I_1, I_2, I_3, \ldots, I_m)$, each cluster (j) corresponds to a vector $W_j = (W_{j1}, W_{j2}, W_{j3}, \ldots, W_{jm})$ of adaptive weight. The number of possible clusters $n(j = 1, 2, \ldots, n)$ is arbitrary. The Fuzzy ART weight vector is equivalent to one vector for both ART1 weight vectors *bottom-up, top-down*, that is, subsumes both vectors. Each ART system includes a field, F_0, of nodes that represents a current input vector I^T; a field, F_1, that receives both bottom-up input from F_0 and top-down input from a field, F_2, that represents the active code, or cluster. The F_1 activity vector is denoted **x**=$(x_1, \ldots, x_m)$ and the F_2 activity vector is denoted **y**=$(y_1, \ldots, y_n)$. The number of nodes in each field is arbitrary.

1. *Initialize*
   (a) Initially, each cluster is said to be $uncommitted$, after a category is selected for coding it becomes $committed$, and the weight vector W_j is set as $W_{j1}(0) = W_{j2}(0) = \ldots = W_{jm}(0) = 1$
   (b) Then, a choice parameter α, a learning rate β, and a vigilance parameter ρ are set: $\alpha > 0, \beta \in [0, 1], \rho \in [0, 1]$.
2. *Complement Coding*
   (a) To improve the reliability of category choice, input a is expanded with complement. $I = (a, a^c)$, $a^c = 1 - a$.
3. *Category choice*
   (a) For each I and cluster (F_2 node) j, the choice function T_j is defined by $T_j(I) = \frac{|I \wedge W_j|}{\alpha + |W_j|}$, where the fuzzy AND operator $\wedge$ is defined by $(x \wedge y)_i = min(x_i, y_i)$ and $|\ |$ is defined by $|x| = \sum_{i=1}^{m} |x_i|$
   (b) The system makes a cluster selection when more than one cluster could be selected at a given time. The index J denotes the chosen cluster, where

$$T_J = max\{T_j : j = 1, \ldots, n\} \tag{2}$$

If more than one T_J is maximal, the system chooses the category with the smallest j index. Nodes become $committed$ in order $j = 1, 2, 3, \ldots$.

4. *Resonance or Reset*

   (a) The resonance occurs if the match function of the chosen cluster meets the vigilance criterion; where

$$\frac{|I \wedge W_J|}{|I|} \geq \rho \tag{3}$$

   The learning processes is done according to the equation

$$W_J^{(new)} = \beta(I \wedge W_J) + (1 - \beta)W_J^{(old)}. \tag{4}$$

   Fast learning corresponds to $\beta = 1$, which is the learning rule in Eq. 4

$$W_J^{(new)} = \beta(I \wedge W_J) \tag{5}$$

   (b) *Mismatch reset* occurs if $\frac{|I \wedge W_J|}{|I|} < \rho$.

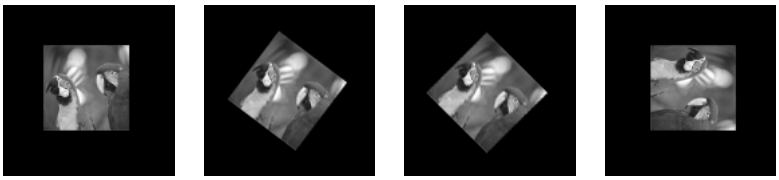
   Then the value of the choice function T_J is set to 0 for the duration of the input presentation to prevent the persistent selection of the same category selection during search. A new index J is chosen by (2). The search continues until the chosen J satisfies (3). If no one of the clusters is selected, then a new cluster must be incremented in $F2$ field.

4 Experimental Results

Three different data sets are generated. The first data set consists of 128x128 bits monochromatic images representing the digits from 0 to 9 in Arial font. The second and third data sets consist of 13 different texture classes. The images of the 13 classes were collected from the web site Vistex[1] texture database.

For the first and second data sets, 50 different images are generated for each image considering different orientation, scale and translation. So, the database consists of ten rotations of seven degrees ($0°, 7°, \ldots, 63°$) and five scaled versions (0.5, 0.8, 1, 1.5 and 2), we have 50 different images for each class. In the first database we have 10 different classes, so at all, the first database consists of 500 images and the second database has 13 classes and a total of 650 images. Finally, the third data set consists of similar geometric transformations occurred in the second data set. Four noisy versions of each of the 10 rotation versions (also $7°$ rotation). Patterns with Gaussian noise are used for training. The noise distribution is of mean $\mu = 1$ and standard deviations of $\sigma = 0.005$, 0.01 and 0.15, generating a total of 520 images.

The tests made with the considered model aim at to evaluate the quality of the extracted invariant characteristics. We used binary and gray-level images, and they suffered geometric transformations as it can be seen in Figure 5. The modifications done to every pattern in the data set were as follow: rotation angles were $37°, 45°, 90°$.

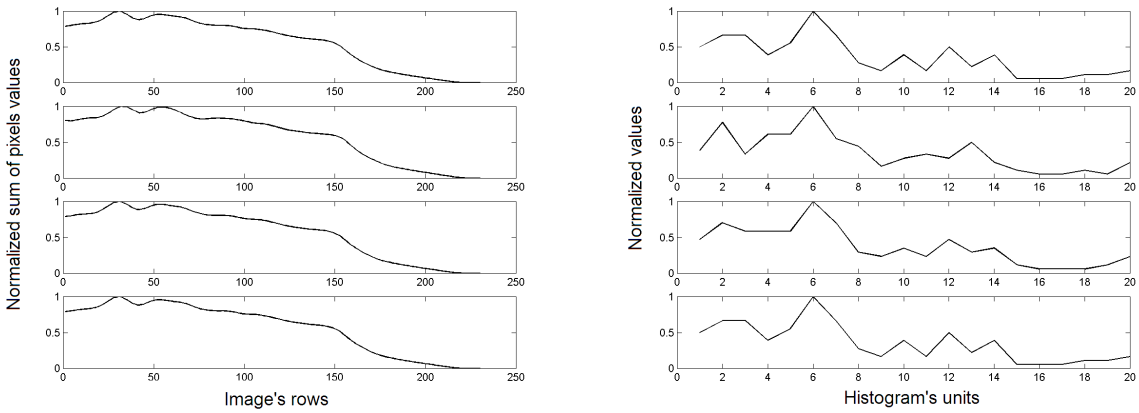


$0°$ $37°$ $45°$ $90°$

Figure 5: Geometric transformations of the "parrots" image. From left to right, rotation angles are $0°$, $37°$, $45°$ and $90°$, respectivately.

We used two vectors for invariant classification. The first one, named "horizontal scan vector", is a normalized vector; and the second one, named "vertical scan vector", remember that this vector is not rotational invariant. So, the

[1](http://vismod.media.mit.edu/vismod/imagery/imagery.html)



(a) (b)

Figure 6: Characteristics vector of "parrots" image, (a) horizontal scan vector, and (b) histogram of vertical scan vector with rotations of $0°$, $37°$, $45°$ and $90°$ degrees.

second step is to calculate the histogram of the "vertical scan vector". After some tests we found that a histogram of 20 elements was satisfactory for classification.

Figure 6(a) Shows 4 normalized horizontal scan vectors of "parrots" images with different rotations. In the case of $90°$ degrees rotation the vector is the same as the original image's vector ($0°$ rotation). In the case of $37°$ and $45°$, horizontal scan vectors have little difference in relation to the original image's vectors. This happens because of the discrete characteristics of digital images, i.e., when the image is rotated new pixels are added and some pixels are lost. Despite the little differences, vectors still conserve similar characteristics, allowing these vectors to be classified in the same cluster. Similar characteristic is observed in vertical scan vector's histogram, Figure 6(b).

Figure 7 presents horizontal and vertical scan vectors for noisy images, with mean $\mu = 0$ and standard deviation of $\sigma = 0.05, 0.1$ and 0.15. The first vector of the image is noiseless, similarity among this vector and the noisy vector decays as the noise is incremented. Again, these similarities are acceptable for the classifier, so we can conclude that our model presents a certain level of noise tolerance.

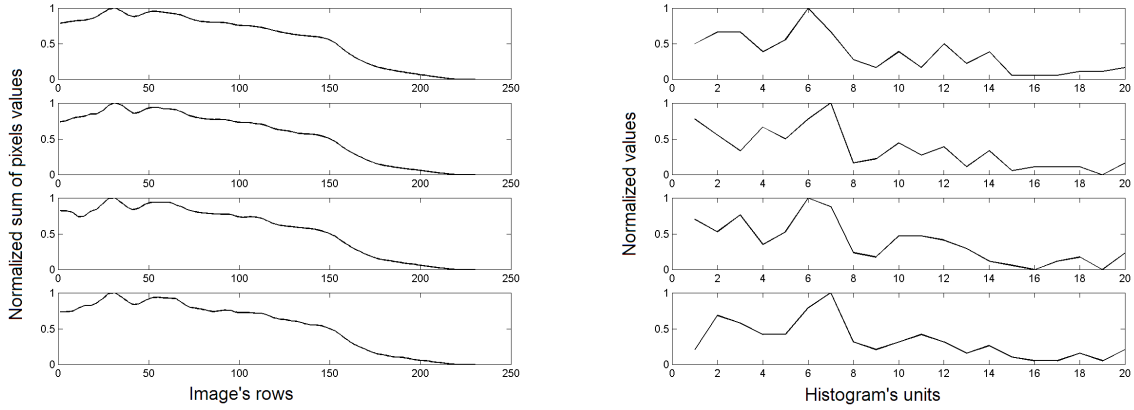


Figure 7: Horizontal scan vectors and vertical scan vector's histograms of noisy images with $\mu = 0$ and σ = 0.05, 0.10 and 0.15.

The selected parameters for experiments are described in Table 1(a).

The performance of our proposed model on different data sets is shown in Table 1(b). We compare the results of our technique with the results of Zernike moments (ZM) [9, 8]. The classification accuracy achieved in this work was

	Proposed model			ZM	
	Binary	Texture	Noisy	Binary	Texture
α	10	100	0.5	40	30
β	1	1	1	1	1
ρ	0.73	0.89	0.86	0.95	0.82

(a)

	Proposed Model	ZM
Binary	99.85%	90.2%
Texture	99.39%	68.6%
Noisy	88,88%	–

(b)

Table 1: (a) Parameters values for *fuzzy* ART. (b) Classification results of data sets.

satisfactory, binary data set 99.85%, texture data set 99,39%, and noisy data set 87,88%. Patterns with bigger density noise than we used in this work were classified in different clusters.

5 Conclusions

The proposed model can efficiently recognize patterns regardless of their possible position, rotation and scale variations. This model consists of two systems: feature detection and classification system. The former works as invariant feature extractor while the latter acts as a robust classifier. We can conclude that the proposed features set and the accompanying features selection method are effective for pattern classification problem.

6 Acknowledgements

The authors are grateful to CNPq and CAPES for the financial support of this work.

References

[1] A.K. Jain and R.P.W. Duin and J. Mao : Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence. **22(1)** (2000) 4–37.

[2] D. Ballard and C. Brown : Computer Vision. Prentice-Hall. (1982)

[3] O. Trier and A. K. Jain and T. Taxt: Feature extraction methods for character recognition. Pattern Recognition. **29(4)** (1996) 641–662

[4] E. Barnard and D. Casasent : Invariance and Neural Nets. IEEE Trans. on Neural Network. **2(5)** (1991) 498–508.

[5] M.K. Hu : Pattern Recognition by Invariant Moments. Proc. IRE Transactions on Information Theory. (1961) 179–187.

[6] G. Carpenter, S. Grossberg : A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. Computer Vision, Graphics and Image Processing. **37** (1987) 54–115.

[7] G. Carpenter, S. Grossberg, D. B. Rosen : Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. Neural Network. **4(6)** (1991) 759–771.

[8] A. Khotanzand and J-H. Lu : Classification of invariant image representations using a neural network. IEEE Transactions on Acoustic, Speech, and signal Processing. **38(6)** (1990) 1028–1038.

[9] C. Kan and M.D. Srinath : Combined features of cubic B-spline wavelet moments and Zernike moments for invariant pattern recognition. International Conference on Information Technology: Coding and Computing. (2001) 511-515.

[10] Mara Lúcia M. Lopes, Carlos R. Minussi and Anna Diva P. Lotufo : Electric load forecasting using a fuzzy ART&ARTMAP neural network. Applied Soft Computing, **5(2)** (2005) 235–244.

Constraint Solving Using Dynamic Variable and Value Selection

Carlos Castro, Eric Monfroy[*], Christian Figueroa, and Rafael Meneses
Universidad Técnica Federico Santa María
Valparaíso, Chile
FirstName.Name@inf.utfsm.cl

Abstract

The notion of strategy is very important to obtain an efficient resolution in Constraint Programming. When solving a specific problem, one has to make some a priori choices to statically determine the "best" strategy. However, the effects and efficiencies of strategies are generally unpredictable. In this work, we propose to dynamically change a strategy when it is showing bad performances, trying to avoid computing too long with bad strategies. Our interest is to propose a general framework that could assure good performances without needing the know-how of experts, allowing the use of Constraint Programming technology by a wider range of people. Some first experimental results show the effectiveness of our framework and approach.

Keywords: constraint programming, constraint propagation, adaptive strategy.

Resumen

En el ámbito de la Programación con Restricciones, la noción de estrategia es muy importante para realizar una resolución eficiente. Cuando se está resolviendo un problema específico uno debe hacer algunas elecciones a priori para determinar estáticamente la "mejor" estrategia. Sin embargo, en general, los efectos y la eficiencia de las estrategias son impredecibles. En este trabajo, se propone cambiar dinámicamente una estrategia cuando ésta está mostrando malos comportamientos, tratando de evitar cómputos demasiado largos con malas estrategias. El interés de este trabajo es proponer un marco general que pueda asegurar buenos rendimientos sin necesidad del conocimiento de expertos, permitiendo el uso de la tecnología de la Programación con Restricciones por parte de un rango de personas más amplio. Resultados experimentales preliminares muestran la eficiencia de este marco y de este enfoque.

Palabras claves: Programación con restricciones, propagación de restricciones, estrategias adaptativas.

1 INTRODUCTION

A Constraint Satisfaction Problem (CSP) is defined by a set of variables, a set of values for each variable, and a set of constraints. The goal is to find an instantiation (or all instantiations) of values for each variable such that the set of constraints is satisfied. This kind of problem is currently solved using both complete and incomplete techniques. Specifically, the Constraint Programming community uses a complete approach that can be seen as an interleaving process between enumeration and domain reduction. Enumeration is carried out by classical backtracking algorithm, and domain reduction consists in eliminating values that cannot participate in any solution to the set of constraints. Concerning domain reduction a lot of research has been done on local consistency techniques and it is generally accepted that arc-consistency and bound consistency are good choices taking into account effort and narrowing of the search space. On the other hand, when enumerating two decisions have to be made: selection of the variable to be enumerated and selection of the value to enumerate.

Some efforts have been done in order to evaluate general criteria, such as Minimum Domain or Maximum Degree for variable selection, and Lower Bound, Upper Bound or Splitting for value selection. Indeed, for some applications, such as Job-Shop Scheduling problems, specific variable and value selection have been

[*]also LINA, Université de Nantes, France

proposed [6]. Some previous and current works study static criteria –the selection criteria is determined just once before the beginning of the solving process and it remains unchanged during the whole process– or dynamic criteria –the selection criteria are changed during the solving process–. Anyway, it is well-known that an a priori decision concerning a good variable and value selection before solving the CSP is very hard since strategy effects are rather unpredictable. We could say that the a priori decision of the best criteria is almost impossible in the general case. Some efforts have also been done in order to determine the best strategy based on information coming from the solving process [15, 5, 2, 3, 8]. In those works the analysis is always carried out previous to the solving process and trying to detect the best strategy.

Similarly to adaptive constraint satisfaction of [4], we are interested in dynamically detecting bad decisions concerning variable and value selections: instead of trying to predict the effect of a strategy, we evaluate the efficiency of running strategies, and replace strategies showing bad results. Indeed, we want the solvers to decide when to continue applying a given criteria or to change for applying another criteria. To this end, we first define measures of the solving process that are observed accordingly to some time-based or computation-based policies. Then, this information is analysed to update the priority of application of each strategy: we penalise strategies that have done a bad work, and we give more credits to those that we judge efficient. Finally, we select the strategy to apply next based on these priorities. These three processes –observation, analysis, update– are repeated until a solution is found or an unsatisfiable problem is detected. Some preliminary results show the effectiveness of our framework in terms of solving efficiency. Although we focus on complete methods (based on reduction and enumeration) the framework we propose could also be adapted for taking into account strategies for constraint propagation, incomplete solvers, or even hybrid solvers [7, 13]. Note that our approach is orthogonal to techniques that try to predict good strategies (e.g., [12, 9]), and thus, can be combined with such techniques.

This paper is organised as follows: Section 2 gives the motivations for doing this work, Section 3 presents our general approach whereas Section 4 presents an instantiation for finite domains constraints. Section 5 reports experimental results obtained with our implementation of dynamic strategies for finite domains. Finally, we conclude the paper in Section 6.

2 MOTIVATIONS

Solvers based on constraint propagation alternate phases of constraint propagation (pruning of the search space) with phases of enumeration (branching). Although all strategies of enumeration are valid (they do not lose solution), their effects on resolution efficiency is very difficult to predict (and in most cases, impossible). Moreover, no strategy is the best (or one of the best) for all classes of problems. In this section, for a given problem, we show the importance of selecting a good strategy, or avoiding a bad one when detected.

An enumeration strategy consists in selecting a variable and one value of its domain to perform enumeration. In order to better explain the importance of variable and value selection criteria, we have solved several instances of the classical N-Queens problem. Each instance has been solved using the nine strategies obtained by the combination of three variable selection criteria (the variable with smallest domain, with largest domain, and with an average domain) and three value selection criteria (the upper bound, the lower bound, and the middle value of the domain. These strategies will be detailed later in Section 3.

In Tables 1 and 2, we thus consider 9 solvers, each one using a different enumeration strategy. For example, $S_{\downarrow\updownarrow}$ is the solver that performs enumeration by always selecting the variable with the smallest domain, and taking the middle value of its domain. These tables where obtained using the constraint programming system Oz [1].

In each column of Table 1, we determine the best solver in terms of time, and give it the value 1. The others are expressed as a ratio with respect to the best solver, e.g., n meaning that the strategy is n times slower than the best one. We use a time limit of 10 minutes before stopping a solver: a run that was stopped is represented by a "-". For small problems (10 or 20 queens), the strategy is not very relevant. But for larger ones, the three first strategies appear to be better suited for this class of problems. There is a ratio of about 100 between the best and worst solvers that found a solution in less than 10 minutes. For each column, the best solver computes the first solution in about a handful of milliseconds [2]. This means that after more than 10000 times the CPU time required by the best solver the solvers with bad strategies could

[1] The system Mozart can be obtained from `http://www.mozart-oz.org`

[2] There can be several 1 in a column: times being too small to be compared properly.

queens	10	20	50	100	150	200
$S_{\downarrow\downarrow}$	1	1	1	1	84	84
$S_{\downarrow\updownarrow}$	1	1	1	1	1	1
$S_{\downarrow\uparrow}$	1	1	1	1	78	33
$S_{\updownarrow\downarrow}$	1	1	1	–	–	–
$S_{\updownarrow\updownarrow}$	1	1	–	–	2	–
$S_{\updownarrow\uparrow}$	1	1	1	–	–	–
$S_{\uparrow\downarrow}$	1	82	–	–	–	–
$S_{\uparrow\updownarrow}$	1	1	–	–	–	–
$S_{\uparrow\uparrow}$	1	82	–	–	–	–

Table 1: Ratio of CPU time

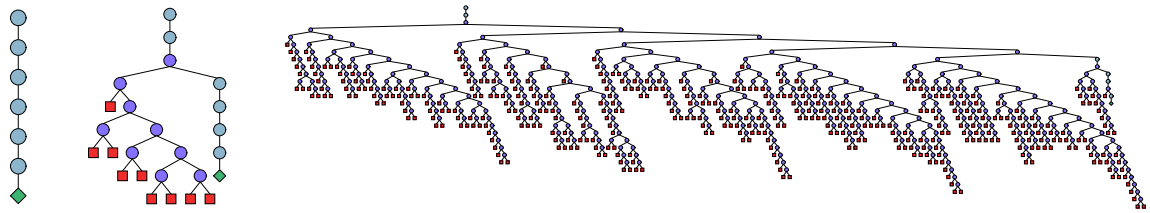


Figure 1: 10-Queens solved with 3 strategies: $S_{\downarrow\updownarrow}$, $S_{\downarrow\uparrow}$, and $S_{\uparrow\uparrow}$

not find a solution. This motivates the fact that it is very important to detect a bad strategy and to replace it by a better one.

queens	10		20		50		100		150		200	
	e	*b*	*e*	*b*	*e*	*b*	*e*	*b*	*e*	*b*	*e*	*b*
$S_{\downarrow\downarrow}$	24	17	76	60	1065	1021	137	41	638131	637978	292640	292449
$S_{\downarrow\updownarrow}$	6	0	41	27	56	11	110	13	152	7	207	9
$S_{\downarrow\uparrow}$	24	17	76	60	1065	1021	137	41	636270	636119	291313	291121
$S_{\updownarrow\downarrow}$	103	96	75	59	2165	2120	–	–	–	–	–	–
$S_{\updownarrow\updownarrow}$	13	5	1847	1827	–	–	–	–	4952	4815	–	–
$S_{\updownarrow\uparrow}$	103	96	75	59	2165	2120	–	–	–	–	–	–
$S_{\uparrow\downarrow}$	820	807	646606	646561	–	–	–	–	–	–	–	–
$S_{\uparrow\updownarrow}$	6	1	251153	251127	–	–	–	–	–	–	–	–
$S_{\uparrow\uparrow}$	820	807	650407	650375	–	–	–	–	–	–	–	–

Table 2: Number of enumerations and backtracks

Table 2 illustrates the number of basic operations performed during resolution. A cell contains 2 numbers e and b representing the total number of enumerations (e), from which how many enumerations where not worth, i.e., backtracks (b).

Again we see the effect of strategies. Let us have a closer look at the 20 queens problem. Whereas one of the fastest solver achieved 41 enumerations and 27 backtracks, the worst one finding a solution required more than 600000 enumerations and more than 600000 backtracks. Again we have a ratio of more than 10000 between solvers. This is still worse when we compare a solver that terminates with a solution and one that does not, for example, for the 200 queen problem: the solver $S_{\uparrow\uparrow}$ was stopped after more than 10000 times more operations than the operations required by the solver $S_{\downarrow\updownarrow}$ to find a solution.

Figure 1 shows three search trees, each one corresponding to the resolution of the 10-queens problem with different strategies. We can see that the first strategy ($S_{\downarrow\updownarrow}$) directly goes to a solution (6 enumerations and no backtrack). The second one ($S_{\downarrow\uparrow}$), after a bad choice for the second enumeration (generating 17

backtracks), finally goes directly to a solution. The last strategy ($S_{\uparrow\uparrow}$) performs numerous wrong choices (generating 807 backtracks) before getting a solution.

By observing the CPU times, the number of enumerations and backtracks, and some search trees, it is obvious that different strategies have significantly different efficiencies. Thus, it is crucial to select a good strategy; however, this cannot be predicted in the general case.

In this paper, we are interested in observing strategies in order to detect bad strategies and replace them by better ones. We are thus concerned with meta strategies, i.e., building a solver dynamically using several basic strategies: our solver uses a single strategy at a time; but when a strategy is judged to have a poor behaviour, it is replaced by another one in order to explore differently the search tree. We thus do not try to predict the best strategy, but we try to obtain a fairly good strategy. Although observing resolution is an overhead, this is negligible compared to the significant differences between strategies.

3 A GENERIC APPROACH

In this work, we focus on computing just one solution for a CSP: on one hand, we guess that it will be easier in a first time to detect bad strategies and, on the other hand, when using Constraint Programming for optimisation a sequence of CSPs is solved searching the first solution for each one; thus, obtaining a first solution quickly is very important. However, our approach can be extended for computing all solutions. It is important to note that in this paper we do not try to compute the best a priori strategy, instead we want to avoid bad and slow strategies. Thus, we observe what happens during computation and decide to change strategies when they become inefficient.

3.1 Overview

The framework we propose here intend to avoid using for a long time a bad enumeration strategy. The framework is based on two processes able to communicate and exchange information.

The first process runs the SOLVE algorithm, a generic solving algorithm which alternates constraint propagation phases with enumeration phases. Moreover, we consider that SOLVE does not have just one enumeration strategy, but it has a set of enumeration strategies. Each strategy is characterised by a priority that changes during computation. SOLVE will use only one strategy at a time, but it can change of strategy during resolution. Thus, SOLVE applies a meta strategy of enumeration, made of a sequence of several basic enumeration strategies. This algorithm also manages exploration of the search space, represented as a search tree: nodes are CSPs reduced by propagation; the root node is the initial CSP after propagation; a node is split in 2 nodes by enumeration; leaves represent either a solution of the CSP or a failure.

The second process runs the ANALYZE algorithm. This algorithm aims at observing some properties of the current CSP, the current search tree, and past computations. These observations are not performed continuously, but at some given moments. We thus call them snapshots, and they can be seen as an abstraction of the resolution at a time t. Using these snapshots, the ANALYZE algorithm evaluates the different strategies, and it can change the priority of each enumeration strategy used in the SOLVE algorithm.

SOLVE is thus devoted to computation (resolution) applying a meta strategy for enumeration which is depending on the evaluations of ANALYZE. We can now detail each component of our framework.

3.2 Basic enumeration strategies

A basic enumeration strategy splits a node of the search tree into two nodes. For example, consider a CSP C. A basic enumeration step on variable x from C creates a choice point with 2 nodes: one is C such that a value v is assigned to x ($x = v$), the other one which considers C with the remaining domain for the variable x ($x \in D \backslash v$). Here we focus on enumeration, but the technique can be extended to splitting a domain into 2 or more sub domains each one with more than one element. A basic enumeration strategy thus selects a variable and a value of this variable. However, in the general case, this is very difficult, if not impossible, to know the effect of the enumeration. There are thus several possible strategies that have been defined empirically.

There are some common selection of variables: the variable with the smallest domain, the most constrained variable (i.e., the one that appears in the largest number of constraints). These two strategies are generally used for first fail techniques (trying to lead to a failure as soon as possible for pruning the

search tree). However, strategies that select a variable with a domain of average size (we call it Med in the following), or select the variable with the largest domain, can turn to be the most efficient for some problems or classes of problems. There are numerous strategies, such as those based on observations of the graph of constraints (e.g., the variable which is a link between two sub problems of a CSP), or those devoted to special computation domains.

Then, the issue is to select a value of the variable: the minimum value of the domain, the maximum value, the middle value, the value that gives the best outcome of the objective function in case of optimisation, shaving, ... Once again, these selections will have totally different effects on the efficiency of the enumeration, e.g., the middle value is well suited for the N-queen problem.

In the following, a basic enumeration strategy is a combination of a variable and a value selection. Note that in general, all combinations are not well suited, or do not define interesting strategies. We also consider that the SOLVE algorithm has at disposal a set $\mathcal{S}$ of enumeration strategies, each one attached to a priority.

3.3 The notion of priority

For each strategy i of $\mathcal{S}$ we define a real number p_i representing the priority of the strategy i. These priorities can be fare, but we can also consider unequal priorities when more knowledge is given: for example, knowing the class of problems, a higher priority can be given to some solvers known as efficient to solve these problems. At each enumeration step, the SOLVE algorithm applies the strategy of highest priority to perform the enumeration. Priorities are updated by the ANALYSE algorithm w.r.t. the work achieved by the strategies.

3.4 The SOLVE algorithm

The sketch of our SOLVE algorithm (Figure 2) is very simple (note that it is close to the SOLVE algorithm of [1]). Given a CSP, it computes the first solution, or proves that there is no solution by exploring all the search space. The SOLVE algorithm is an interleaving process of constraint propagation and enumeration.

```
SOLVE:
        WHILE not solved OR failed
         constraint propagation
         IF not failed AND not solved THEN
                update strategy priority list
                select enumeration
                enumerate
                proceed by case
         END
    END
```

Figure 2: The SOLVE algorithm

Constraint propagation consists in reducing domains of variable by removing locally inconsistent values. We do not detail this phase here as our framework is valid for each kind of propagation (this is done in the next section for a precise use). Roughly speaking, constraint propagation iteratively removes variables by enforcing some level of local consistency, such as arc consistency or path consistency [11]. Following a given strategy such as Forward Checking, enforcing consistency of variables directly linked to the freshly enumerated variable by a constraint, or Look Ahead, enforcing a local consistency to the complete CSP [10].

The select enumeration function composes our meta strategy: it selects the best current basic enumeration strategy with respect to the priorities attached to all basic strategies; our meta strategy is thus the sequence of basic strategies. The enumerate function is the application of the selected basic strategy.

Proceed by case is a function that manages the choice points created by enumeration, and thus enables one to explore the search tree. It can be a depth first, breadth first, ... exploration of the tree.

3.5 Snapshots

A snapshot is an abstraction of a resolution state (a search space and past computations) at a time t. Taking a snapshot consists in extracting information from a resolution step: this is achieved by recording

and measuring certain criteria and properties of the state. Thus, two main issues are important: when taking snapshots, which measures and criteria to take into account.

The first issue is related to the type of problems we are solving, the computation domain, and last but not least, the implementation architecture used for the SOLVE and ANALYSE processes. Snapshots can be taken regularly and mechanically, such as every n milliseconds, or every m loops of the SOLVE algorithm (i.e., every m sequences of propagation + enumeration). But snapshots can also be taken irregularly when some events appear, such as a variable that has been fixed by propagation, the search space that has been reduced of x %, a slow reduction cycle that has been detected, ... In these cases, taking snapshots is more related to what we want to observe. We see here, that a concurrent implementation largely ease taking snapshots related to time or special events.

The measures and criteria taken into account can be of all kinds. They aim at reflecting resolution, and updating priorities of strategies. Thus, they depend on the computation domain, and on knowledge about strategies (e.g., when do strategies are known to perform well or not). Common measures can be the maximum depth of the search tree that has been reached, the depth of the current node, the size of the search space, the number of fixed variables, ... But snapshots can also contain information related to constraints or CSP properties such as a constraint was linearised, a constraint is monotonous, the CSP is triangular in terms of variables, ... Finally, snapshots can reflect also past computation by recording how long was a propagation phase, which operators were used, ...

3.6 The ANALYSE algorithm

In order to be able to take snapshots as wanted (e.g., at each enumeration step, after a given elapsed time, after a given number of steps, ...), we consider that the ANALYSE algorithm and the SOLVE algorithm run concurrently in two different processes. We do not detail here the interaction between these two processes since this is implementation-driven. Roughly, there are two common possibilities: we can consider two processes sharing data (the search tree), or a query/answer system where the update strategy process asks the SOLVE process in order to get the required measures. Roughly, the ANALYSE algorithm (Figure 3) takes the snapshots as required and using its knowledge, it analyses these snapshots, evaluates progress by computing some "differences" between snapshots, and finally it updates the priorities based on the progress evaluation.

ANALYSE:
take a snapshot F_i
compute progress w.r.t. some of the previous snapshots F_js
update strategy priorities w.r.t. the progress and F_i
schedule the next snapshot
END

Figure 3: The ANALYSE algorithm

The basic strategy priorities are updated w.r.t. the work achieved. This is evaluated in terms of progress evaluation and of properties that appear on the taken snapshots.

The progress is composed of some δ_i, i.e., differences of criteria and measures between snapshots. For example, if the number of variables fixed by enumeration was n in the F_{i-1} snapshot, and this number is $n-3$ in F_i, the progress can contain information such as: there was a backtrack of at least three levels. If F_{i+1} is $n-7$, we can deduce that a wrong choice (bad enumeration) was made at least 7 nodes above the node which was being explored at F_{i-1}. If the size of the search space was s in F_{i-1} and it is less than $0.1 * s$ in F_i, the progress will contain that the current enumeration strategy lead to some drastic reduction (by enumeration and propagation).

Properties that appear on the last snapshot can be: the CSP is now triangular, a constraint is now linear, a "hard" variable is now fixed, ...

The knowledge of the ANALYSE algorithm is contain in a set of rules. The head of such a rule is a conjunction of conditions on the deltas and properties, while the body is a conjunction of updates of strategies priorities:

$$cond_1 \wedge \ldots \wedge cond_l \wedge prop_1 \wedge \ldots \wedge prop_k \Rightarrow p_1 = p_1 + \alpha_1 \wedge \ldots \wedge p_m = p_m + \alpha_m$$

where:

- the $cond_i$ are conditions on some deltas, either $\delta_j\{\leq,\geq,=\}constant$ or $\sum\omega_j \times \delta_j\{\leq,\geq,=\}constant$ where ω_j is a positive or negative constant representing the weight of each δ_j in the condition,
- the $prop_i$ are some properties extracted from snapshots: $prop_i(F_{i_1},\ldots,F_{i_k}$ is true if the property i is true on $F_{i_1},\ldots,F_{i_k}$, and
- a $alpha_i$ is a constant to increase or decrease the priority of the strategy i.

When the head is fulfilled (i.e., conditions and properties are verified), the body is executed and thus the priorities are updated. These updates are also communicated to the SOLVE algorithm.

4 A PRACTICAL APPROACH

We now use our generic approach in order to improve finite domain CSP resolution. We simplify the framework and consider some simple snapshots in order to observe few criteria.

We fix the constraint propagation: we consider arc consistency [11] computation with a Forward Checking strategy [10] (i.e., after enumeration of a variable n, only variables linked to n by a constraint are reduced). The proceed by case strategy is a depth first left first exploration of the search tree: from a node, this procedure selects the left hand side child node, i.e., the one that assigns a value to the enumerated variable.

Nine strategies We consider 9 basic enumeration strategies that are the combination of 3 variable selection strategies and 3 value selection strategies. Concerning variables we consider the following well known criteria: selection of the variable with minimum domain (denoted $\downarrow$), the variable with largest domain (denoted $\uparrow$), and the variable with an average domain (denoted $\updownarrow$). Concerning value selection we use the following criteria: selection of the smallest value of the domain (denoted $\downarrow$), the largest value of the domain (denoted $\uparrow$) the middle value of the domain (denoted $\updownarrow$). The enumeration strategy that consists in selecting the variable with the largest domain and the smallest value of its domain will thus be denoted $\uparrow\downarrow$. The solver which always uses this strategy is denoted by $S_{\uparrow\downarrow}$. Our dynamic solvers that can change of strategies will be denoted by S_{Dyn}.

Snapshots The observation we want to make on resolution are related to the used strategies. Moreover, for some first experimentations we decided to observe few criteria and to keep a small snapshot:

- $Maxd$: the maximum depth reached in the search tree,
- d: the depth of the current node,
- s: the size of the current search space,
- f: the percentage of variables fixed by enumeration (in the current node),
- f': the percentage of variables that has been fixed by enumeration or propagation (in the current node),

Analysis and Update The snapshots are taken each n ms, n being a parameter. We only keep track of the two last snapshots F and F^-, and thus we use deltas computed from these 2 snapshots:

- $\delta_1 = Maxd_F - Maxd_{F^-}$ represents a variation of the maximum depth, i.e., whether the search went deeper in the search tree between the two snapshots,
- $\delta_2 = d_F - d_{F^-}$: if δ_2 is positive, the current node is deeper than the one explored at the previous snapshot,
- $\delta_3 = 100 * (s_{F^-} - s_F)/s_{F^-}$ a percentage of reduction since F^-; if positive the size of the current search space is smaller than at snapshot F^-,
- $\delta_4 = f_F - f_{F^-}$: if positive, reflects an improvement in the degree of resolution made by enumeration,

- $\delta_5 = f'_F - f'_{F^-}$: if positive, reflects an improvement in the degree of resolution.

The rules we use for updating priorities are very simple:

$$w_1 * \delta_1 + w_2 * \delta_2 + w_3 * \delta_3 + w_4 * \delta_4 + w_5 * \delta_5 \geq c_1 \Rightarrow p = p + \alpha$$

$$w'_1 * \delta_1 + w'_2 * \delta_2 + w'_3 * \delta_3 + w'_4 * \delta_4 + w'_5 * \delta_5 \leq c_2 \Rightarrow p = p - \beta$$

where

- p is the priority of the currently used strategy,
- w_i and w'_i are positive weights used to favour some deltas, and thus, some criteria observed on the search tree,
- α and β are two positive constants, with $\alpha < \beta$, and
- c_1 and c_2 are 2 constants, with $c_2 < c_1$.

The first rule "rewards" the current strategy because it obtained a score over a given threshold, i.e., it performed well w.r.t. to our criteria. The reward is to give it a higher priority; thus, it will remain the one with the best priority and run for more time. The second rule penalises the running strategy when the work it does is judged inefficient. The priority is decreased. Note that penalties are greater than reward because we want to stop inefficient strategies. Note also that a strategy obtaining a penalty may remain the one with the best priority if it worked efficiently (during several snapshots) before.

5 EXPERIMENTAL RESULTS

Our prototype implementation in Oz uses two processes and a mechanism of query/answer between the processes. The first process runs the SOLVE algorithm, whereas the second one runs the ANALYSE algorithm. In order to be able to extend and modify our system, we did not try to optimise performances: we copy and store lots of contexts in order to track and trace resolution.

The solvers and strategies are names as above: for example, $\downarrow\downarrow$ is the strategy that selects the variable with the smallest domain, and enumerates its smallest value. $S_{\downarrow\downarrow}$ is the solver that always use the strategy $\downarrow\downarrow$.

For 100 and 200 Queens, we take a snapshot every 200 ms, whereas for smaller problems a snapshot is taken every 50 ms. The observations and deltas are the ones described in the previous section. For the update rules, we fixed the w_i and w'_i at 1 (no favour for a special delta), $c_1 = 0$, $\alpha = 0.1$, $c_2 = 0$, and $\beta = 0.2$.

The solver $S_{Dyn\sim}$ uses the following dynamic meta strategy: choose the strategy with the best priority; if two strategies have the same priority, randomly choose one. Each time, we perform several runs of the $S_{Dyn\sim}$ solver since the first selected strategy slightly influence the efficiency, meaning that our solver could benefit of static strategy selection for the beginning.

Table 3 illustrates resolution of some N-Queen problem instances obtained with the 9 solvers with fix strategy, and our dynamic strategy solver $S_{Dyn\sim}$. As in Section 2, for each column 1 represents the fastest of the 9 solvers, and n that this solver is n times slower than the fastest one. The line timeout represents the percentage of timeouts of our dynamic solver (as before, solver are stopped after 10 minutes). $S_{DynAv\sim}$ represents the average of the runs of $S_{Dyn\sim}$, $S_{DynW\sim}$ the worst run that does not reach the timeout, and $S_{DynB\sim}$ the best run. The average line is the average information of the 9 solvers with fixed strategies. Only runs that do not reached the time out are used for the average, and for $S_{DynAv\sim}$.

For small problems, parameters seem to be well suited: there is no timeout, and the average run of $S_{Dyn\sim}$ is better than the average of the solver with fixed strategies. $S_{DynB\sim}$ performs as well as the best of the 9 solvers. For 20-queens, $S_{DynW\sim}$ is far better than the worst solver with fix strategy. For large problems, we do not improve the number of timeouts (we are even a bit worst for 200 queens). We should change parameters for larger problems. But, best cases are the same with dynamic or static strategies. For 200 queens, the average successful runs of $S_{Dyn\sim}$ are far better than the average successful runs of the solvers with static strategies. This is the same for the worst run of $S_{Dyn\sim}$ which is far better than the worst static strategy, and very close to the best static strategy.

queens	10			20			50			100			200		
	r	e	b	r	e	b	r	e	b	r	e	b	r	e	b
$S_{\downarrow\downarrow}$	1	24	17	1	76	60	1	1065	1021	1	137	41	84	292640	292449
$S_{\downarrow\updownarrow}$	1	6	0	1	41	27	1	56	11	1	110	13	1	207	9
$S_{\downarrow\uparrow}$	1	24	17	1	76	60	1	1065	1021	1	137	41	33	291313	291121
$S_{\updownarrow\downarrow}$	1	103	96	1	75	59	1	2165	2120	–	–	–	–	–	–
$S_{\updownarrow\updownarrow}$	1	13	5	1	1847	1827	–	–	–	–	–	–	–	–	–
$S_{\updownarrow\uparrow}$	1	103	96	1	75	59	1	2165	2120	–	–	–	–	–	–
$S_{\uparrow\downarrow}$	1	820	807	82	646606	646561	–	–	–	–	–	–	–	–	–
$S_{\uparrow\updownarrow}$	1	6	1	1	251153	251127	–	–	–	–	–	–	–	–	–
$S_{\uparrow\uparrow}$	1	820	807	82	650407	650375	–	–	–	–	–	–	–	–	–
Average	1	213	205	1	172261	172239	1	1303	1258	1	128	31	39	194720	194526
$S_{DynAv\sim}$	1	123	116	1	6002	5985	1	1060	1016	1	128	31	1	644	452
$S_{DynB\sim}$	1	6	0	1	41	27	1	56	11	1	110	13	1	207	9
$S_{DynW\sim}$	1	820	807	3	25400	25483	1	2054	2013	1	137	41	2	1082	895
timeout	0%			0%			55%			66%			77%		

Table 3: N-Queens: CPU Ratio, number of enumerations, and backtracks

Table 4 shows how many times a basic strategy was selected in the meta strategy of $S_{Dyn\sim}$ to solve the 200-Queens problem instance. Note that between two consecutives snapshots a strategy can be applied several times, but we do not take into account that number.

strategy	$\downarrow\downarrow$	$\downarrow\updownarrow$	$\downarrow\uparrow$	$\updownarrow\downarrow$	$\updownarrow\updownarrow$	$\updownarrow\uparrow$	$\uparrow\downarrow$	$\uparrow\updownarrow$	$\uparrow\uparrow$
$S_{DynB\sim}$	0	3	0	0	0	0	0	0	0
$S_{Dyn\sim}$*slow repair*	87	132	154	85	78	106	379	207	181
$S_{Dyn\sim}$*quick repair*	0	2	0	0	6	0	0	0	0

Table 4: 200-Queens: selection of strategies in various runs of S_{Dyn}

For the best run, only the strategy $\downarrow\updownarrow$ was applied. As seen before, this strategy is the best for 200 queens with a static strategy. Our analyse always judged it well, and thus, when starting computation with it, it is not replaced in the dynamic strategy. The second line shows a run for which many changes of strategies were performed. The computation started badly, and our dynamic meta strategies encountered problems to get a basic strategy that could perform well on the remaining problem. In this case, we had a slow repair of the strategy. Although the efficiency was not very good, the dynamic solver could compute a solution before the timeout, whereas the solver (with fixed strategy) continuing with the same "bad" strategy reached the timeout without solution.

The last row shows a run that was judged well at the beginning with the $\updownarrow\updownarrow$ strategy. However, the strategy was then judged to behave badly, and was replaced by $\downarrow\updownarrow$. Here, we obtain a very quick repair of the strategy. Note that this dynamic strategy runs nearly as fast as the fastest solver with fixed strategy. Note also that the solver using only $\updownarrow\updownarrow$ reached the timeout without solution.

It is important to note that for these 2 last runs, the dynamic strategy (and replacement of strategies) was necessary to reach a solution before the timeout.

6 CONCLUSION

We have presented a preliminary work on automatic variable and value selection for constraint solving. Based on the performance of each combination of variable and value selection our dynamic approach has been able to detect bad cases and then to replace strategies. Results show that it is possible to get good performances without a priori choices. The first experimental results are encouraging and promising. However, we also

see that fixing the parameters (deltas, weights of deltas, threshold, penalties, credits, snapshot frequency, ...) of our framework is crucial: thus we plan to integrate some learning in order to tune these parameters automatically, and to detect what information should be observed.

We are interested in extending our approach to local consistency verification, i.e., dynamically changing the local consistency level, and the strategy for reduction. In this work, we have limited our analysis to past events, however, it could also be interesting to take into account information concerning what remains to be solved. We are also interested in using this framework for developing hybrid solvers involving Constraint Programming and Local Search, or Constraint Programming and Genetic Algorithms. Our framework is well suited for hybrid solvers based on Chaotic Iteration [14]. Concerning selection of solvers some work could be done in order, for example, to fix priorities for some well-known cases where the sequence of solvers can be established a priori.

References

[1] Krzysztof R. Apt. *Principles of Constraint Programming.* Cambridge University Press, 2003.

[2] J. Christophe Beck and Eugene C. Freuder. Simple Rules for Low-Knowledge Algorithm. In *CPAIOR 2004*, pages 50–64, 2004.

[3] J. Christophe Beck, Patrick Prosser, and Richard J. Wallace. Variable Ordering Heuristics Show Promise. In *CP 2004*, pages 711–715, 2004.

[4] James Borrett, Edward Tsang, and Natasha Walsh. Adaptive constraint satisfaction. In *15th UK Planning and Scheduling Special Interest Group Workshop*, Liverpool, 1996.

[5] Tom Carchrae and J. Christophe Beck. Low-Knowledge Algorithm Control. In *AAAI 2004*, pages 49–54, 2004.

[6] Yves Caseau and François Laburthe. Improved clp scheduling with task intervals. In *ICLP '94*, pages 369–383, 1994.

[7] Carlos Castro and Eric Monfroy. Designing Hybrid Cooperations with a Component Language for Solving Optimization Problems. In *AIMSA'04*, volume 3192 of *Lecture Notes in Computer Science*, pages 447–458, September 2004.

[8] Marco Correia and Pedro Barahona. Machine Learned Heuristics to Improve Constraint Satisfaction. In *SBIA 2004*, pages 103–113, 2004.

[9] Pierre Flener, Brahim Hnich, and Zeynep Kiziltan. A meta-heuristic for subset problems. In *Proceedings of Practical Aspects of Declarative Languages, PADL 2001*, volume 1990 of *Lecture Notes in Computer Science*, pages 274–287. Springer, 2001.

[10] Vipin Kumar. Algorithms for Constraint-Satisfaction Problems: A Survey. *Artificial Intelligence Magazine*, 13(1):32–44, 1992.

[11] Alan K. Mackworth. Consistency in Networks of Relations. *AI*, 8:99–118, 1977.

[12] Steven Minton. Automatically configuring constraint satisfaction programs: A case study. *Constraints*, 1(1/2):7–43, 1996.

[13] Eric Monfroy and Carlos Castro. A Component Language for Hybrid Solver Cooperations. In *ADVIS'04*, volume 3261 of *Lecture Notes in Computer Science*, pages 192–202, October 2004.

[14] Eric Monfroy, Frédéric Saubion, and Tony Lambert. On hybridization of local search and constraint propagation. In *ICLP*, pages 299–313, 2004.

[15] Christian Schulte and Peter J. Stuckey. Speeding up constraint propagation. In *CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pages 619–633, September 2004.

Estudio Experimental de un Algoritmo Rápido de Intersección para Secuencias Ordenadas

Ricardo A. Baeza-Yates y Alejandro J. Salinger
Centro de Investigación de la Web
Departamento de Ciencias de la Computación
Universidad de Chile
Blanco Encalada 2120
Santiago 6511224, Chile
{rbaeza,asalinge}@dcc.uchile.cl

Abstract

This work presents an experimental comparison of intersection algorithms for sorted sequences, including the recent algorithm of Baeza-Yates. This algorithm performs on average less comparisons than the total number of elements of both inputs (n and m respectively) when $n = \alpha m$ ($\alpha > 1$). We can find applications of this algorithm on query processing in Web search engines, where large intersections, or differences, must be performed fast. In this work we concentrate in studying the behavior of the algorithm in practice, using for the experiments test data that is close to the actual conditions of its applications. We compare the efficiency of the algorithm with other intersection algorithm and we study different optimizations, showing that the algorithm is more efficient than the alternatives in most cases, especially when one of the sequences is much larger than the other.

Keywords: Set operations, merging, multiple search, Web search engines, inverted indices.

Resumen

Este trabajo presenta una comparación experimental de algoritmos de intersección para dos secuencias ordenadas, incluido el reciente algoritmo de Baeza-Yates. Este algoritmo realiza en promedio menos comparaciones que el número total de elementos de ambas entradas (n y m, respectivamente) cuando $n = \alpha m (\alpha > 1)$. Las aplicaciones de este algoritmo se encuentran en el procesamiento rápido de consultas en los motores de búsqueda Web, donde intersecciones o diferencias de conjuntos deben calcularse rápidamente. En este trabajo nos concentramos en estudiar el comportamiento del algoritmo en la práctica, utilizando para los experimentos datos de prueba cercanos a los datos reales de sus aplicaciones. Se comparó la eficiencia del algoritmo con otros algoritmos de intersección y se estudiaron diferentes optimizaciones, mostrando que es más eficiente que las alternativas en la mayoría de los casos, especialmente cuando una secuencia es mucho mayor que la otra.

Palabras claves: Operaciones sobre conjuntos, *merging*, búsqueda múltiple, motores de búsqueda Web, índices invertidos.

1. Introducción

Este trabajo estudia distintos algoritmos para calcular la intersección de secuencias ordenadas. Este problema constituye un caso particular de un problema genérico llamado búsqueda múltiple [3] (ver también [18], problema de investigación 5, página 156), el cual consiste en, dado un multiconjunto D (es decir, puede tener elementos repetidos), de n elementos escogidos de un universo que posee una relación de orden total, buscar en D cada elemento de otro multiconjunto Q, de m elementos escogidos del mismo universo. Los elementos encontrados constituyen, justamente, la intersección de ambos conjuntos.

El problema de intersección de listas ordenadas encuentra su motivación en los motores de búsqueda Web, pues la mayoría de ellos utiliza índices invertidos, en los cuales para cada palabra diferente se tiene una

lista de documentos o posiciones donde ella aparece. Generalmente estas listas están ordenadas por algún criterio, ya sea por posición, un ranking global precalculado, frecuencias de ocurrencias en un documento, u otro. Para calcular el resultado de una consulta, en la mayoría de los casos de requiere intersectar estas listas. En la práctica estas listas total o parcialmente ordenadas pueden tener cientos de millones de elementos, por lo que resulta útil contar con un algoritmo que es rápido y eficiente en promedio.

En el caso en que D y Q son conjuntos (y no multiconjuntos) ordenados, la búsqueda múltiple puede resolverse mezclando (*merging*) ambos conjuntos. Sin embargo, esto no resulta óptimo en todos los casos. Por ejemplo, si m es pequeño (digamos si $m = o(n/\log n)$), es mejor hacer m búsquedas binarias, obteniéndose un algoritmo de tiempo $O(m \log n)$ [2], siendo la métrica de complejidad el número de comparaciones entre cualquier par de elementos. El algoritmo de Baeza-Yates [1] alcanza ambas complejidades dependiendo del valor de m. En promedio realiza menos de $m + n$ comparaciones cuando ambos conjuntos están ordenados, considerando ciertas suposiciones pesimistas.

Este trabajo se enfoca en el estudio experimental de este último algoritmo y de diferentes optimizaciones a éste. Los experimentos consistieron en medir el tiempo de ejecución del algoritmo original y sus optimizaciones con listas de números enteros aleatorios ordenados y compararlos con un algoritmo basado en *merge* y con el algoritmo *Adaptive*, propuesto por Demaine *et al.* [11], el cual parece ser el más utilizado en la práctica. Nuestros resultados muestran que el algoritmo de Baeza-Yates es ligeramente mejor que Adaptive y mucho mejor que Merge cuando los tamaños de las listas difieren bastante.

En la sección 2 presentamos el trabajo relacionado. La sección 3 presenta la motivación para nuestro problema y algunas consideraciones prácticas. La sección 4 presenta el algoritmo de intersección de Baeza-Yates y algunas optimizaciones propuestas. La sección 5 presenta los resultados experimentales obtenidos. En este artículo supondremos que $n \geq m$ y los logaritmos son en base dos a menos que se indique explícitamente otra base. Una versión más sucinta y revisada de este trabajo será presentada en inglés en [7].

2. Trabajo Relacionado

Para resolver el problema de determinar si cualquier elemento de un conjunto de $n + m$ elementos es igual a otro se requieren a lo menos $\Theta((n + m) \log(n + m))$ comparaciones en el peor caso (ver [14]). Sin embargo, esta cota inferior no se aplica al problema de búsqueda múltiple ni, equivalentemente, al problema de intersección de conjuntos. Inversamente, las cotas inferiores del problema de búsqueda sí se aplican al problema de unicidad de un elemento [13]. Explotando esta idea, Demaine *et al.* definieron un algoritmo adaptativo de intersección múltiple de conjuntos [11, 12]. También definieron la dificultad de una instancia del problema, la cual fue redefinida luego por Barbay y Kenyon [9].

Cuando los conjuntos se encuentran ordenados las cotas inferiores para la intersección constituyen a su vez cotas inferiores para mezclar ambos conjuntos. Sin embargo, lo inverso no es cierto, pues en la intersección no necesitamos encontrar la posición real de cada elemento en la unión de ambos conjuntos, sólo debemos determinar si es que un elemento está en el otro conjunto o no. Existe bastante trabajo sobre el problema de mezcla con un número mínimo de comparaciones en el peor caso. Sin embargo, casi no se ha realizado trabajo sobre el caso promedio, pues éste no hace mucha diferencia. Esto último no es cierto para el problema de búsqueda múltiple, y por lo tanto para la intersección [3].

En el caso de la mezcla, Fernández de la Vega *et al.* [16] analizaron el caso promedio de una versión simplificada de la mezcla binaria de Hwang-Lin [17] encontrando que si $\alpha = n/m$ con $\alpha > 1$ y no potencia de 2, entonces el número esperado de comparaciones es

$$\left(r + \frac{1}{1 - \left(\frac{\alpha}{\alpha+1}\right)^{2^r}} \right) \frac{n}{\alpha}$$

con $r = \lfloor \log_2 \alpha \rfloor$. Cuando α es potencia de 2, el resultado es más complicado, pero similar. Fernández de la Vega *et al.* [15] también diseñaron un algoritmo probabilístico que mejora el algoritmo de Hwang-Lin en el peor caso para $1{,}618m \leq n \leq 3m$.

El algoritmo de Baeza-Yates [1] intenta adaptarse a los valores de la entrada. En el mejor caso el algoritmo realiza $\lceil \log(m + 1) \rceil \lceil \log(n + 1) \rceil$ comparaciones, lo cual para $m = O(n)$, es $O(\log^2(n))$. En el peor caso, el número de comparaciones que realiza el algoritmo es

$$W(m, n) = 2(m + 1) \log((n + 1)/(m + 1)) + 2m + O(\log(n)) \ .$$

Entonces, para m pequeño, el algoritmo es $O(m \log(n))$ y es $O(n)$ si $n = \alpha m$. En este caso, la razón entre este algoritmo y *merging* es $2(1 + \log(\alpha))/(1 + \alpha)$ asintóticamente, siendo 1 cuando $\alpha = 1$. El peor caso es peor que *merging* para $1 < \alpha < 6,3197$, teniendo su máximo cuando $\alpha = 2,1596$, donde es $1,336$ veces más lento que *merging*. Por lo tanto el peor caso del algoritmo coincide con la complejidad de ambos enfoques, *merging* y búsqueda binaria múltiple, adaptándose a m. Para el caso promedio, bajo suposiciones pesimistas se demuestra que el número de comparaciones es

$$A(m, n) = (m + 1)(\ln((n + 1)/(m + 1)) + 3 - 1/\ln(2)) + O(\log n) \ .$$

Para $n = \alpha m$, la razón entre este algoritmo y *merging* es $(\ln(\alpha) + 3 - 1/\ln(2))/(1 + \alpha)$, lo cual es a lo sumo 0,7913 cuando $\alpha = 1,2637$ y 0,7787 cuando $\alpha = 1$. Este algoritmo es detallado en la sección 4.

3. Motivación: Proceso de Consultas en Índices Invertidos

Los índices invertidos se utilizan en la mayoría de los sistemas de recuperación de texto [4]. Lógicamente, éstos son un vocabulario (conjunto de palabras únicas encontradas en el texto) y una lista de referencias por palabra a sus ocurrencias (típicamente un identificador de documento y una lista de posiciones de la palabra en cada documento). En sistemas simples (modelo Booleano), las listas se ordenan por identificador de documento, y no existe *ranking* (es decir, no hay noción de relevancia de un documento). En ese escenario, el algoritmo básico de intersección se aplica directamente para calcular operaciones Booleanas sobre los identificadores de documentos: la unión (OR) equivale a *merge*, la intersección (AND) es la operación bajo estudio (sólo buscamos los elementos comunes), y la diferencia implica eliminar los elementos que están en el otro conjunto, lo que es similar a una intersección. En la práctica, las listas no se ordenan secuencialmente, sino por bloques. Sin embargo, estos bloques son de gran tamaño, y las operaciones pueden realizarse bloque a bloque.

En sistemas complejos se utiliza un *ranking*. Un *ranking* se basa típicamente en estadísticas de palabras (número de ocurrencias de una palabra por documento y la inversa del número de documentos que la incluye). Ambos valores pueden precalcularse y las listas de referencias se ordenan luego en orden decreciente por frecuencia de las palabras en el documento para así tener primero los documentos más relevantes. Las listas son luego procesadas en orden decreciente del inverso del número de documentos en los cuales cada una aparece (es decir, procesamos primero las listas más cortas), para así obtener primero los documentos más relevantes. Sin embargo, en este caso no siempre podemos tener una asignación de identificadores de documentos tal que las listas queden ordenadas según ese orden. Sin embargo, están parcialmente ordenadas por identificador para todos los documentos que tienen igual frecuencia de palabra.

El esquema anterior fue inicialmente utilizado en la Web, pero a medida que la Web fue creciendo, el *ranking* fue deteriorándose pues las estadísticas de palabra no siempre representan el contenido y calidad de una página Web y, además, puede ser engañado repitiendo y agregando palabras (casi) invisibles. En 1998, Page y Brin [10] describieron un motor de búsqueda (el cual fue el punto de partida de Google) que utilizaba enlaces (*links*) para medir la calidad de una página (PageRank). A esto se le llama *ranking* global basado en popularidad, y es independiente de la consulta realizada. Está fuera del alcance de este trabajo explicar Pagerank, pero podemos decir que éste modela un usuario aleatorio de la Web y que el *ranking* de una página es la probabilidad de que el usuario la visite. Esta probabilidad induce un orden total que puede utilizarse como identificador de documento. Por lo tanto, en un motor de búsqueda basado solamente en enlaces podemos utilizar el algoritmo de intersección tal como antes. Sin embargo, actualmente se utilizan esquemas de *ranking* híbridos que combinan evidencia de enlaces y palabras. A pesar de esto, un *ranking* basado en enlaces entrega buenos resultados aproximando bien el *ranking* real (que puede ser corregido mientras se calcula).

Otro tipo importante de consulta es la búsqueda de frases. En este caso utilizamos la posición de la palabra para saber si una palabra sigue o precede a otra. Entonces, como las frases suelen ser cortas, después de encontrar las páginas Web que contienen todas sus palabras, podemos procesar las dos primeras palabras[1] para encontrar pares adyacentes y luego ellas con la tercera y así sucesivamente. Esto es como calcular una intersección particular donde en vez de encontrar elementos repetidos tratamos de encontrar elementos correlativos (i e $i + 1$), y como las posiciones de las palabras están ordenadas, podemos utilizar nuevamente un algoritmo de intersección. Lo mismo es cierto para búsquedas por proximidad. En este caso, podemos

[1] En realidad es más eficiente utilizar las dos palabras con listas más cortas, y así sucesivamente hasta llegar a la lista más larga si es que la intersección aún no es vacía.

tener un rango k de posibles posiciones válidas (es decir $i \pm k$) o podemos utilizar un *ranking* diferente dependiendo de la proximidad.

Finalmente, en el contexto de la Web, un algoritmo adaptativo es mucho más rápido en la práctica pues la suposición de una distribución uniforme es pesimista. En la Web, la distribución de las ocurrencias de las palabras es bastante desigual. Lo mismo ocurre con la frecuencia de las consultas. Ambas distribuciones siguen una ley de Zipf generalizada[2] [4, 6]. Sin embargo, la correlación entre ambas distribuciones es media [8] a baja [5]. Esto implica que el largo promedio de las listas involucradas en la consulta no es tan desbalanceado. Cuando la correlación es 0, si los largos promedio de las listas, n y m, cuando se toma una muestra, satisfacen $n = \Theta(m)$ (uniforme), en vez de $n = m + O(1)$ (ley de Zipf). Sin embargo, en ambos casos el algoritmo bajo estudio contribuye a una mejora.

4. El Algoritmo de Intersección

A continuación describimos el algoritmo de Baeza-Yates. Supongamos que D está ordenado. En este caso, si Q es pequeño, conviene buscar cada elemento de Q en D. Ahora, cuando Q también está ordenado, la intersección puede resolverse con *merging*. Tanto en el peor caso como en el caso promedio, una mezcla directa necesita $m + n - 1$ comparaciones. Sin embargo, para la intersección de conjuntos se puede hacer algo mejor. El siguiente algoritmo, que se basa en una doble búsqueda binaria, constituye una mejora sobre *merge* en el caso promedio bajo suposiciones pesimistas.

El algoritmo funciona de la siguiente manera. Primero buscamos con búsqueda binaria la mediana (elemento del medio) de Q en D. Si se encuentra, agregamos este elemento al resultado. Encontrado o no, hemos dividido el problema en buscar los elementos menores que la mediana de Q a la izquierda de la posición encontrada en D, o en la que debería estar el elemento si es que no se encontró, y los elementos mayores que la mediana hacia la derecha de esa posición. Luego resolvemos recursivamente en ambos pares de segmentos utilizando el mismo algoritmo. Si en cualquier momento, el tamaño del subconjunto de Q que estamos considerando es mayor que el del subconjunto de D, intercambiamos los roles de Q y D. Notar que la intersección de conjuntos es simétrica en este sentido. Si cualquiera de los subconjuntos es vacío, no se hace nada. La figura 1 muestra el algoritmo en pseudo-código.

```
Intersect(D, Q, minD, maxD, minQ, maxQ)
1.  //si es que Q o D es vacío, termina la recursión
2.  if  minD > maxD or minQ > maxQ
3.      return φ
4.  mediaQ ← Q[posMediaQ]
5.  posMediaD ← bbinaria(mediaQ, D, minD, maxD)
6.  if D[posMediaD] == mediaQ
7.      Resultado ← Resultado ∪ {mediaQ}
8.  if |D[minD..posMediaD − 1]| > |Q[minQ..posMediaQ − 1]| //el subconjunto de D es mayor que el de Q
9.      Resultado ← Resultado ∪ Intersect(D, Q, minD, posMediaD − 1, minQ, posMediaQ − 1)
10. else //se cambian los papeles de D y Q
11.     Resultado ← Resultado ∪ Intersect(Q, D, minQ, posMediaQ − 1, minD, posMediaD − 1)
12. if |D[posMediaD + 1..maxD]| > |Q[posMediaQ + 1..maxQ]|//el subconjunto de D es mayor que el de Q
13.     Resultado ← Resultado ∪ Intersect(D, Q, posMediaD, maxD, posMediaQ + 1, maxQ)
14. else //se cambian los papeles de D y Q
15.     Resultado = Resultado ∪ Intersect(Q, D, posMediaQ + 1, maxQ, posMediaD, maxD)
16. return Resultado
```

Figura 1: Algoritmo de doble búsqueda binaria para intersección de dos conjuntos ordenados.

El algoritmo presentado en [1] puede verse como una versión balanceada del algoritmo de Hwang y Ling [17] adaptado a nuestro problema.

Una optimización simple al algoritmo es aplicar el algoritmo original no sobre ambos conjuntos D y Q completos, sino que buscar el subconjunto de uno de los dos donde existe un traslape, y por lo tanto, donde realmente se pueden encontrar elementos que formen parte de la intersección.

[2]Esta ley indica que el $i-$ésimo elemento tiene una frecuencia proporcional a $1/i^{\alpha},\ \alpha > 0$.

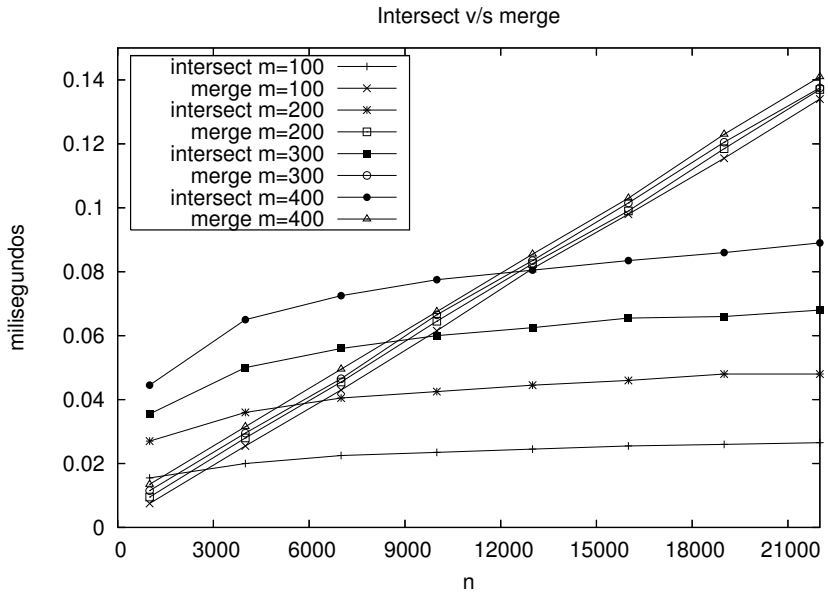


Figura 2: Resultados experimentales para intersect y merge para distintos valores de n y m

Se comienza comparando el menor elemento de Q con el mayor de D, y el mayor de Q con el menor de D. Si es que ambos conjuntos no se traslapan, la intersección es vacía. De lo contrario, se buscan el menor y el mayor elemento de D en Q para encontrar el traslape, lo que toma sólo tiempo $O(\log(m))$. Luego se aplica el algoritmo original con D y el subconjunto encontrado de Q. Esto mejora tanto el caso promedio como el peor caso. El caso en que se busca también el traslape en D también es válido, sin embargo esto toma tiempo $O(\log(n))$ y puede resultar costoso cuando m es mucho menor que n. Esta optimización se menciona en [1], pero no se estudia si es efectiva o no.

5. Resultados Experimentales

Comparamos la eficiencia de nuestro algoritmo, el cual llamamos *Intersect* con una implementación de intersección utilizando *merge* y con una adaptación del algoritmo *Adaptive* [11, 12] para la intersección de dos listas. Esta sección muestra, además, los resultados de los experimentos realizados con las optimizaciones al algoritmo.

Las listas utilizadas contienen números enteros aleatorios uniformemente distribuidos en el rango $[1, 10^9]$. Se varió el largo de una de las listas (n) de 1.000 a 22.000 con un paso de 3.000. Para cada uno de los largos se intersectaron esas listas con listas de cuatro largos distintos (m), de 100 a 400. Cada punto representa el promedio de la ejecución sobre 20 instancias distintas, y de 1.000 ejecuciones con cada una (para eliminar variaciones debidas al sistema operativo dado los pequeños tiempos resultantes).

Los programas fueron implementados en C utilizando el compilador Gcc 3.3.3 y ejecutados sobre una plataforma Linux con un procesador Intel(R) Xeon(TM) de 3.06GHz con 512 Kb de cache y 2Gb RAM.

La figura 2 muestra una comparación entre nuestro algoritmo y *merge*. Se puede ver que nuestro algoritmo resulta mejor que *merge* a medida que aumenta n y que sus tiempos aumentan considerablemente a medida que aumenta m.

La figura 3 muestra los resultados obtenidos al implementar la optimización del algoritmo Intersect. Para esta comparación, también se agregó a *merge* el cálculo del traslape de ambas listas.

Podemos observar que no existen grandes diferencias entre el algoritmo original y el optimizado, más aún, el algoritmo original resultó algo más rápido que el optimizado. La razón por la cual la optimización no se traduce en una mejora considerable puede radicar en la distribución de los datos de prueba. Al ser los números sacados aleatoriamente de una distribución uniforme, en la mayoría de los casos el traslape de

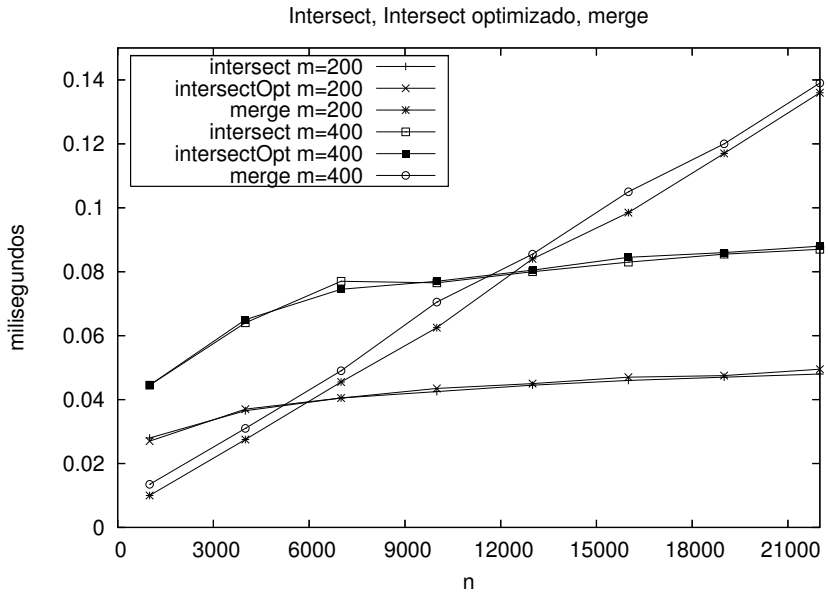


Figura 3: Resultados experimentales para Intersect, Intersect optimizado y merge, para distintos valores de n y para $m = 200$ y $m = 400$

ambos conjuntos abarca gran parte de Q. Entonces, la optimización no produce mejoras y sólo se traduce en que se ocupa algo de tiempo en buscar el traslape.

5.1. Algoritmos Híbridos

A partir de los resultados experimentales podemos ver que existe un tramo de valores de n para los cuales *merge* es mejor que nuestro algoritmo. Surge entonces la idea de combinar ambos algoritmos en un sólo algoritmo híbrido que ejecute cada uno de ellos según resulte conveniente.

Para saber cuál es el punto de corte donde conviene utilizar un algoritmo en vez del otro, para cada valor de n se midió el tiempo de ambos algoritmos con distintos valores de m hasta identificar el valor de m para el cual *merge* resulta más rápido que Intersect. Estos valores de m forman una recta en función de n, la que se observa en la figura 4. Esta recta es aproximada por $m = 0{,}033n + 8{,}884$, con una correlación $r^2 = 0{,}999$.

El algoritmo híbrido consiste en llamar a *merge* cuando $m > 0{,}033n + 8{,}884$, y a Intersect de lo contrario. La condición se evalúa en cada momento de la recursión.

Al modificar el algoritmo el punto de corte cambia y lo que nos gustaría sería encontrar el algoritmo híbrido óptimo. Aplicando la misma idea nuevamente, se encontró la recta que define los valores para los cuales *merge* resulta mejor que el algoritmo híbrido. Esta recta se aproxima por $m = 0{,}028n + 32{,}5$, con $r^2 = 0{,}992$. Definimos entonces el algoritmo Híbrido2 que ejecuta *merge* cuando $m > 0{,}028n + 32{,}5$ e Intersect en caso contrario. Por último, se combinaron ambos híbridos, creando una tercera versión en la cual la recta de corte entre *merge* e *Intersect* es el promedio entre las rectas de los híbridos 1 y 2. La recta resultante es $m = 0{,}031n + 20{,}696$. La figura 4 muestra la recta de corte entre el algoritmo original y *merge* y los resultados obtenidos con los algoritmos híbridos. El algoritmo óptimo en teoría sería el Híbrido.i cuando i tiende a infinito.

Se puede observar que los algoritmos híbridos registran tiempos menores a los del algoritmo original en el tramo en que éste es más lento que *merge*. Sin embargo, en el tramo posterior el algoritmo original resulta más rápido que los híbridos, debido a que en la práctica debemos evaluar el punto de corte en cada paso de la recursión. Entre los algoritmos híbridos, podemos ver que el primero es levemente mejor que el segundo y éste a su vez mejor que el tercero. Una idea para reducir el tiempo en el tramo en que el algoritmo original registra menores tiempos que los híbridos es crear un nuevo algoritmo híbrido que ejecute *merge* cuando resulte conveniente y que luego ejecute el algoritmo original, sin evaluar la relación entre m y n para ejecutar

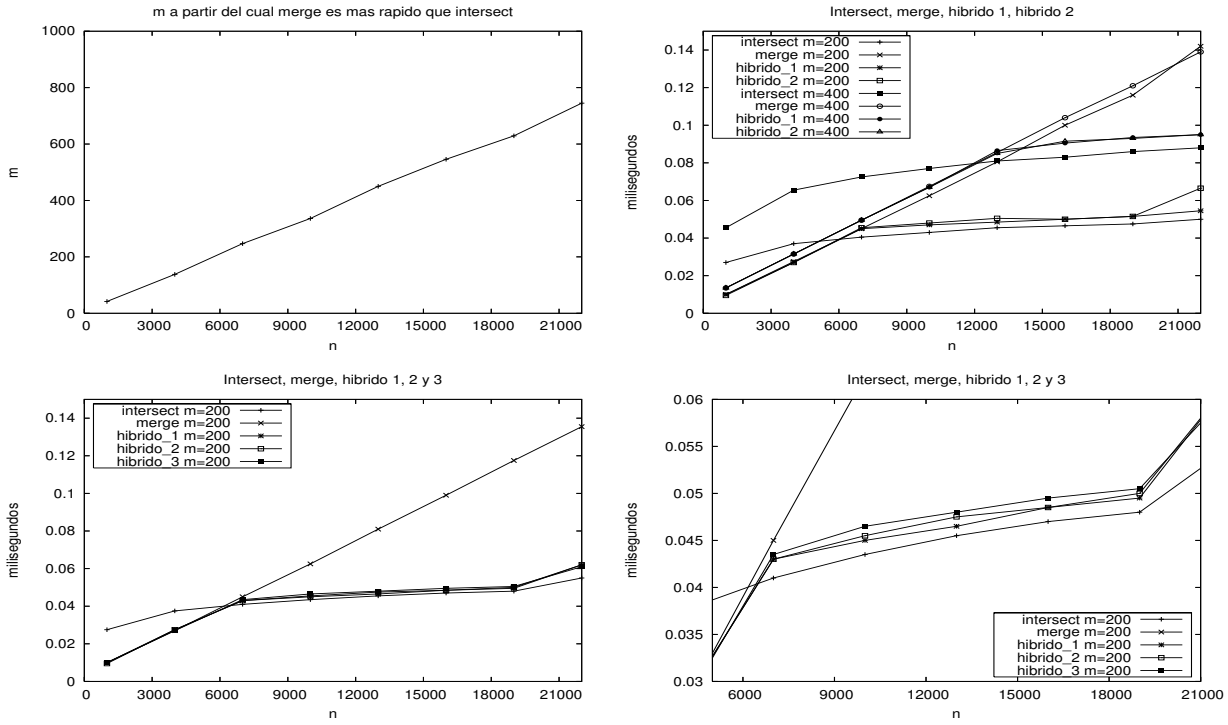


Figura 4: Arriba a la izquierda, valor de m a partir del cual *merge* es más rápido que Intersect. A la derecha, comparación entre el algoritmo original, *merge* y los híbridos 1 y 2 para $m = 200$ y $m = 400$. Abajo, comparación entre Intersect, *merge* y los tres híbridos para $m = 200$. El gráfico de la derecha es un acercamiento del de la izquierda.

merge. Este algoritmo registra el mismo tiempo que Intersect en el tramo donde éste resulta mejor que *merge*, combinando de la mejor manera las ventajas de ambos algoritmos. La figura 5 muestra los resultados obtenidos con este nuevo algoritmo híbrido.

5.2. Algoritmo Adaptive

Comparamos nuestro algoritmo con una versión del algoritmo *Adaptive* [11, 12], el cual funciona de la siguiente manera: se toma uno de los conjuntos, y se elige su primer elemento, el cual llamaremos *elim*. Se busca *elim* en el otro conjunto, haciendo saltos exponenciales, esto es, en las posiciones $1, 2, 4, \ldots, 2^i$. Si es que nos pasamos, es decir, el elemento en la posicion 2^i es mayor que *elim*, hacemos una búsqueda binaria entre las posiciones 2^{i-1} y 2^i. Este tipo de búsqueda en un rango no acotado también es $O(\log n)$. Si encontramos *elim*, se agrega este elemento al resultado. Luego, se marca la posición anterior a la posición donde estaba *elim* (o la posición donde debería haber estado) para acordarnos que desde allí hacia atrás ya se revisó el conjunto. Ahora se elige *elim* como el menor elemento del conjunto que sea mayor al *elim* anterior y se intercambian los roles, buscando ahora en el primer conjunto, haciendo saltos desde la posición que indica lo que ya hemos revisado del conjunto. Si en algún momento no existe un elemento mayor que el que se estaba buscando, se termina.

La figura 6 muestra una comparación entre los tiempos de nuestro algoritmo y *Adaptive*. Se puede observar que los tiempos de ambos algoritmos siguen la misma tendencia y que Intersect resulta mejor que *Adaptive*.

5.3. Largos de las Secuencias con Distribución de Zipf

Una de las aplicaciones del algoritmo, como se dijo anteriormente, está en la búsqueda de documentos en la Web, en la que se cumple que el número de documentos en que aparecen las palabras sigue una distribución

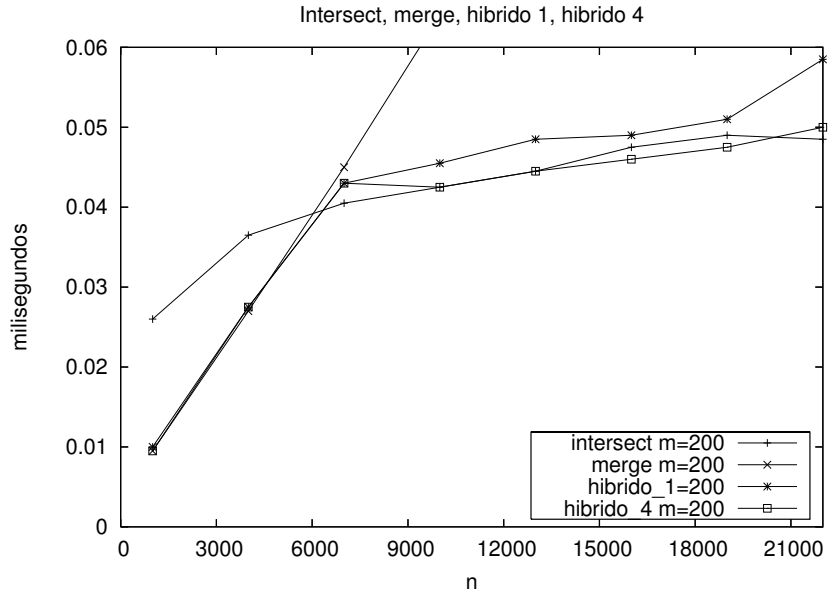


Figura 5: Resultados experimentales para Intersect, merge y los algoritmos híbridos 1 y 4 para distintos valores de n y para $m = 200$

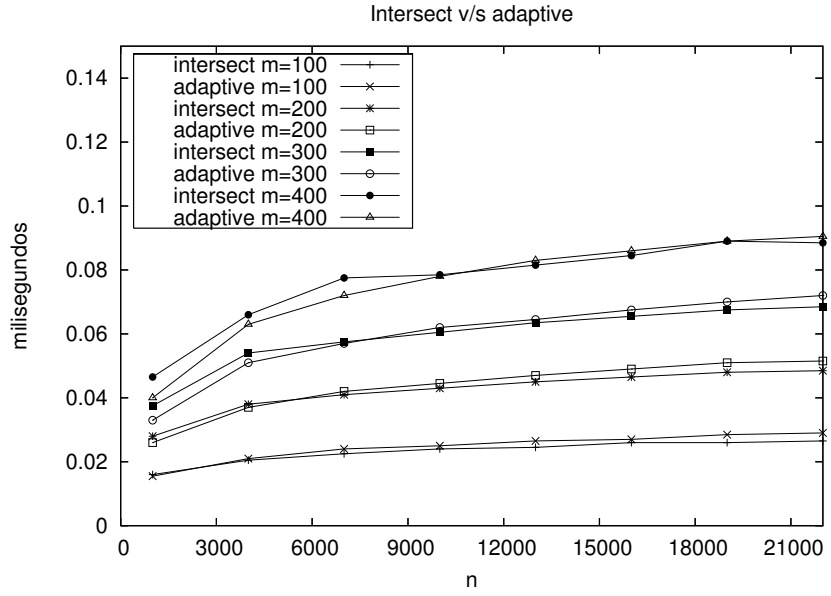


Figura 6: Resultados experimentales para Intersect y Adaptive, para distintos valores de n y m.

de Zipf.

Resulta interesante observar cómo se comporta el algoritmo *intersect* dependiendo de la razón de los largos de las dos secuencias cuando estos largos siguen una distribución de Zipf y la correlación entre ambos conjuntos es cero (caso ideal). Para hacer esta medición, se tomaron dos números aleatorios a y b con distribución uniforme entre 0 y 1.000, con los cuales se calcularon los largos de las listas D y Q como $n = K/a^{\alpha}$ y $m = K/b^{\alpha}$, respectivamente, con $K = 10^9$ y $\alpha = 1{,}8$, asegurando que $n > m$. Se hicieron 1.000 mediciones tomando 80 listas distintas para cada una de ellas, y repitiendo las ejecuciones 1.000 veces para cada lista.

La figura 7 muestra los tiempos obtenidos por ambos algoritmos en función de n/m tanto en escala normal como logarítmica.

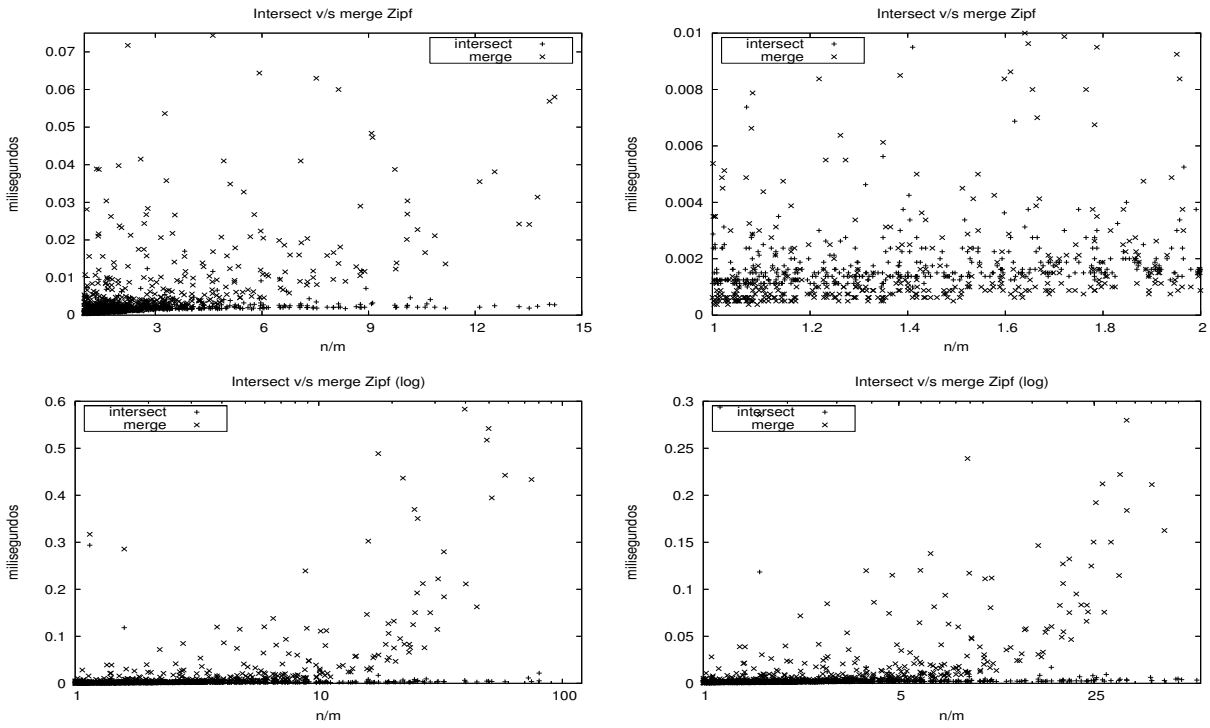


Figura 7: Arriba: tiempos de Intersect y *merge* en función de la razón de los largos de las listas cuando éstos siguen una distribución de Zipf. El gráfico de la derecha es un acercamiento del gráfico de la izquierda. Abajo: tiempos de Intersect y *merge* en escala logarítmica. El gráfico de la derecha es un acercamiento del de la izquierda.

En estos gráficos podemos observar que los tiempos de *Intersect* son menores que los de *merge* cuando n es bastante mayor que m. A medida que la razón entre n y m se hace menor, ya no es tan claro cuál de los dos algoritmos resulta más eficiente. Cuando $n/m < 2$ en la mayoría de los casos los tiempos de *merge* son los menores.

6. Conclusiones

En este trabajo hemos estudiado experimentalmente un algoritmo simple de intersección de conjuntos ordenados que funciona muy bien en promedio y que no inspecciona todos los elementos involucrados. A partir de los experimentos observamos que nuestro algoritmo original resulta más eficiente que *merge* cuando el tamaño de una de las listas es varias veces el tamaño de la otra. Esta mejora se hace más evidente a medida que n aumenta. A su vez, nuestro algoritmo supera a *Adaptive* [11, 12] para cualquier relación entre los tamaños de las listas. El algoritmo híbrido que combina *merge* y nuestro algoritmo a partir de la información empírica obtenida, aprovecha las ventajas de cada uno de ellos y resultó ser el más eficiente.

En la práctica no necesitamos calcular el resultado completo de la intersección de dos listas, pues la mayoría de la gente sólo mira los primeros resultados de una consulta, usualmente menos de las dos primeras páginas de resultados [6]. Además, calcular el resultado completo puede resultar demasiado costoso si es que una de las palabras de la consulta aparece en varios millones de documentos, como ocurre en la Web. Es por esto que, en general, se utilizan evaluaciones parciales y el resultado se va completando a medida que el usuario lo requiere. Si utilizamos el clásico algoritmo de mezcla obtenemos naturalmente primero las páginas Web más relevantes. Con nuestro algoritmo no es tan simple, pues aunque procesemos primero el lado izquierdo del problema recursivo, no necesariamente obtendremos el resultado en el orden deseado. Una solución simple es procesar el conjunto de menor tamaño de izquierda a derecha realizando búsquedas binarias en el conjunto mayor. Sin embargo, esta solución resulta eficiente sólo para pequeños valores de m, alcanzando una complejidad de $O(m \log n)$ comparaciones. Una variante optimista puede usar una predicción del número de páginas del resultado y utilizar un esquema intermedio adaptativo que divide los conjuntos de menor tamaño en trozos no simétricos con una inclinación hacia el lado izquierdo. Resulta entonces interesante estudiar la mejor manera de calcular resultados parciales de manera eficiente.

Como la correlación entre ambos conjuntos en la práctica es entre 0.2 y 0.6 dependiendo del texto Web usado (distribución de Zipf con α entre 1.6 y 2.0) y las consultas (distribución de Zipf con α menor, por ejemplo 1.4), queremos extender nuestros resultados experimentales a este caso. Sin embargo, ya vimos que en ambos extremos (correlación 0 o 1), el algoritmo en estudio es competitivo.

Referencias

[1] R. Baeza-Yates. A Fast Set Intersection Algorithm for Sorted Sequences. En *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM 2004)*, Springer LNCS 3109, pp 400-408, Istanbul, Turkey, July 2004.

[2] R. Baeza-Yates. *Efficient Text Searching*. PhD thesis, Dept. of Computer Science, University of Waterloo, May 1989. Also as Research Report CS-89-17.

[3] R. Baeza-Yates, P.G. Bradford, J.C. Culberson, y G.J.E. Rawlins. The Complexity of Multiple Searching, unpublished manuscript, 1993.

[4] R. Baeza-Yates y B. Ribeiro-Neto, *Modern Information Retrieval*, ACM Press/Addison-Wesley, England, 513 pages, 1999.

[5] R. Baeza-Yates y Felipe Saint-Jean. A Three Level Search Engine Index bases in Query Log Distribution. En *SPIRE 2003*, Springer LNCS, Manaus, Brazil, October 2003.

[6] Ricardo Baeza-Yates. Query Usage Mining in Search Engines. En *Web Mining: Applications and Techniques*, Anthony Scime, editor. Idea Group, 2004.

[7] R. Baeza-Yates y A. Salinger. Experimental Analysis of a Fast Intersection Algorithm for Sorted Frequences. Lecture Notes in CS, Springer, SPIRE 2005, Buenos Aires, November 2005.

[8] R. Baeza-Yates, Carlos Hurtado, Marcelo Mendoza y Georges Dupret. Modeling user search behavior. En *LA-WEB 2005*, IEEE CS Press, Buenos Aires, Noviembre, 2005.

[9] Jérémy Barbay y Claire Kenyon. Adaptive Intersection and t-Threshold Problems. En *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 390-399, San Francisco, CA, January 2002.

[10] S. Brin y L. Page. The anatomy of a large-scale hypertextual Web search engine. En *7th WWW Conference*, Brisbane, Australia, April 1998.

[11] Erik D. Demaine, Alejandro López-Ortiz, y J. Ian Munro. Adaptive set intersections, unions, and differences. En *Proceedings of the 11th Annual ACM-SIAM Symposium, on Discrete Algorithms*, pages 743-752, San Francisco, California, January 2000.

[12] Erik D. Demaine, Alejandro López-Ortiz, y J. Ian Munro. Experiments on Adaptive Set Intersections for Text Retrieval Systems. En *Proceedings of the 3rd Workshop on Algorithm Engineering and Experiments*, LNCS, Springer, Washington, DC, January 2001.

[13] Paul Dietz, Kurt Mehlhorn, Rajeev Raman, y Christian Uhrig. Lower Bounds for Set Intersection Queries. En *Proceedings of the 4th Annual Symposium on Discrete Algorithms*, 194-201, 1993.

[14] David Dobkin y Richard Lipton. On the Complexity of Computations Under Varying Sets of Primitives, *Journal of Computer and Systems Sciences*, 18, 86-91, 1979.

[15] W. Fernández de la Vega, S. Kannan, y M. Santha. Two probabilistic results on merging, *SIAM J. on Computing* 22(2), pp. 261-271, 1993.

[16] W. Fernández de la Vega, A.M. Frieze, y M. Santha. Average case analysis of the merging algorithms of Hwang and Lin. *Algorithmica* 22 (4), pp. 483-489, 1998.

[17] F.K. Hwang y S. Lin. A Simple algorithm for merging two disjoint linearly ordered lists, *SIAM J. on Computing* 1, pp. 31-39, 1972.

[18] Gregory J.E. Rawlins. *Compared to What?: An Introduction to the Analysis of Algorithms*, Computer Science Press/W. H. Freeman, 1992.

A Modification of the Landau-Vishkin Algorithm Computing Longest Common Extensions via Suffix Arrays*

Rodrigo César de Castro Miranda†
Universidade de Brasília, Mestrado em Informática,
Brasília, Brazil, 70910-900
rodrigo.miranda@acm.org

and

Mauricio Ayala-Rincón‡
Universidade de Brasília, Departamento de Matemática,
Brasília, Brazil, 70910-900
ayala@mat.unb.br

Abstract

Approximate string matching is an important problem in Computer Science. The standard solution for this problem is an $O(mn)$ running time and space dynamic programming algorithm for two strings of length m and n. Landau and Vishkin developed an algorithm which uses suffix trees for accelerating the computation along the dynamic programming table and reaching space and running time in $O(nk)$, where $n > m$ and k is the maximum number of admissible differences. Suffix trees are used for pre-processing the sequences allowing an $O(1)$ running time computation of the longest common extensions between substrings. We present a variation of the Landau-Vishkin algorithm which instead of suffix trees uses suffix arrays for computing the longest common extensions.

Keywords: Approximate string matching, suffix trees, suffix arrays.

1 Introduction

Matching strings with errors is an important problem in Computer Science, with applications that range from word processing to text databases and biological sequence alignment. The standard algorithm for approximate string matching is a dynamic programming algorithm with $O(mn)$ running-time and space complexity, for strings of size m and n.

Landau and Vishkin [12] developed an $O(kn)$ algorithm for matching a pattern to a string of length n with at most k differences. The algorithm iterates through the diagonals of the dynamic programming table and uses a suffix tree data structure for constant-time jumps along the diagonals, bypassing character-by-character matching. Despite being of theoretical interest, the Landau-Vishkin algorithm has the practical drawback in that the use of suffix trees implies a large space usage [11], which makes it uninteresting for the treatment of long length sequences which occur in molecular biology.

In this paper we present a variation of the Landau-Vishkin algorithm which instead of suffix trees uses suffix arrays enhanced with a table of longest common prefixes [1, 13] for computing the necessary jumps along the diagonals of the dynamic programming table. Since in contrast with the space usage of suffix trees suffix arrays use less space [15, 10], by this proposed modification the space usage of the Landau-Vishkin algorithm is decreased obtaining a method which is closer to practical usage for the long sequences which arise in molecular processing.

*Work supported by CNPq grant 471791/2004-0.
†Presentation supported by FINATEC.
‡Partially supported by the Brazilian research council CNPq.

Initially, section 2 defines the problem and presents the dynamic programming solution. Afterward, section 3 presents the Landau-Vishkin algorithm, suffix trees and their use in the algorithm and section 4 presents suffix arrays and describes how they could be used instead of suffix trees in the Landau-Vishkin algorithm. Finally, section 5 concludes and remarks on further work.

2 Problem Definition

In this section we will define the problem being studied and present the standard dynamic programming solution.

2.1 Preliminaries

Given the strings $T = t_1...t_n$ and $P = p_1...p_m$ of length $|T| = n$ and $|P| = m$, $m \leq n$, over an alphabet Σ we present a few definitions.

- ε is the empty string.
- P is a *substring* of T if $m \leq n$ and $p_1...p_m = t_i...t_{i+m-1}$ for some $i \geq 1$ and $i + m - 1 \leq n$. If $m < n$ we say that P is a *proper substring* of T.
- P is a *prefix* of T if $m \leq n$ and $p_i = t_i$ for $1 \leq i \leq m$. If $m < n$ then we say that P is a *proper prefix* of T.
- P is a *suffix* of T if $p_1...p_m = t_i...t_{i+m-1}$ for $i + m - 1 = n$ and $i \geq 1$. If $i > 1$ then we say that P is a *proper suffix* of T. We also say that $T_i = t_i...t_n$ where $i \geq 1$ is the *i-th suffix* of T (that is, the suffix of T that starts at position i).
- The *longest common prefix* of T and P is the longest string $L = l_1...l_k$ such that $0 \leq k \leq m$ and $l_1 \dots l_k = p_1 \dots p_k = t_1 \dots t_k$. If $k = 0$ (i.e. P and T do not have a common prefix) then $L = \varepsilon$.
- The *longest common extension* of T and P at position (i, j) is the length of the longest common prefix of T_i and P_j.

In this article we call the string T the *text* and the string P the *pattern*.

2.2 Approximate String Matching

Definition 1 (Edit distance) *The edit distance between two strings $P = p_1...p_m$ and $T = t_1...t_n$ is the minimum amount of operations needed to transform P into T or T into P, where the allowed operations are defined as follows.*

- *Substitution* when a character p_i of P is replaced with a character t_j of T.
- *Insertion* when a character p_i of P is inserted at position j of T.
- *deletion* when a character p_i is removed from P.

The sequence of operations needed to transform P into T is called the *edit transcript* of P into T.

An *alignment* of P and T is a representation of the operations applied on P and T, usually placing one string on top of the other, and filling with a dash ('-') the positions in P and T where a space was inserted so that every character or space on either string is opposite a unique character or unique space on P or T [5].

Definition 2 (Approximate string matching with k differences) *We define the approximate string matching problem with k differences between a pattern P and a text T to be the problem of finding every pair of positions (i, j) in T where the edit distance between P and $t_i...t_j$ is at most k. The special case when $k = 0$ is the problem of finding all occurrences of P in T.*

2.3 Dynamic Programming Solution

We can find the edit distance $D(i,j)$ between $p_1...p_i$ and $t_1...t_j$ from the distances $D(i-1,j-1)$ between $p_1...p_{i-1}$ and $t_1...t_{j-1}$, $D(i-1,j)$ between $p_1...p_{i-1}$ and $t_1...t_j$ and $D(i,j-1)$ between $p_1...p_i$ and $t_1...t_{j-1}$ by solving the following recurrence relation.

$$D(i,j) = \begin{cases} i+j & \text{if } j=0 \text{ or } i=0, \\ min[D(i-1,j-1)+d, D(i-1,j)+1, D(i,j-1)+1] & \text{otherwise} \\ \text{where } d=0 \text{ if } p_i = t_j \text{ or } 1 \text{ if } p_i \neq t_j & \end{cases}$$

This relation can be calculated by a straightforward $O(nm)$ dynamic programming algorithm using an $n+1 \times m+1$ dynamic programming table. A technique due to Hirschberg [6, 5] can be applied to it in order to decrease the space usage to $O(n)$ at the cost of doubling the computation time.

3 The Landau-Vishkin Algorithm

Landau and Vishkin presented an $O(kn)$ algorithm for the approximate string matching problem with k-differences in [12]. The algorithm is divided in two phases: a pre-processing phase and an iteration phase. In the iteration phase the he algorithm iterates k times over each diagonal of the dynamic programming table and finds all matches of P that end at each diagonal with at most k differences. The explanation given follows the one given by Gusfield in [5].

3.1 Diagonals and d-paths

Given the text $T = t_1...t_n$ and the pattern $P = p_1...p_m$, we present a few definitions.

- A *diagonal* d of the dynamic programming matrix D consists of all $D_{i,j}$ such that $j-i=d$.
- The *main diagonal* is the diagonal 0 of D composed by the cells $D_{i,i}$ where $0 \leq i \leq m \leq n$.
- A *path* in the dynamic programming table is a sequence of adjacent cells.
- If a cell (i,j) follows a cell $(i-1,j-1)$ in a path, it is said to be a *mismatch* if $t_j \neq p_i$, and a *match* otherwise.
- If a cell $(i+1,j)$ or a cell $(i,j+1)$ follows a cell (i,j) in a path, it is said to be a *space* .
- A *error* in a path is either a mismatch or a space.
- A *d-path* in D is a path which starts either at column 0 before row $d+1$ or at row 0 and has the following properties:
  - Paths initiating at row 0 start with zero errors and paths initiating at cell $(i,0)$ for $1 \leq i \leq d$ start with i errors (insertion of i spaces before T for the prefix $p_1...p_i$).
  - From any cell (i,j), the next cell along the path (if any) can only be one of $(i+1,j+1)$, $(i,j+1)$, or $(i+1,j)$.
  - It has a total of d errors.
- A d-path is *farthest-reaching* on diagonal i if it is a d-path that ends at cell (r,c) in diagonal i and the index of its ending column c is greater than or equal the index of the ending column of every other d-path which ends on diagonal i.

3.2 d-path Construction and Extension

A d-path is constructed in D in the following way.

If $d=0$ then a 0-path that starts on diagonal i is a path that begins at cell $D_{0,i}$ and is extended along diagonal i to cell $D_{j,i+j}$, where j is the length of the longest common prefix of $t_{i+1}...t_{i+j}$ and $p_1...p_j$.

A d-path starting at cell $(d,0)$ for $0 < r \leq m$ is extended in the same way as a 0-path from row 0.

If $d>0$ then a d-path is constructed from a $(d-1)$-path whose ending cell $D_{r,c}$ is on diagonal i by firstly extending it to cell $D_{r',c'}$ in diagonal i' in one of three ways:

- the path is extended one cell to the right to cell $D_{r',c'} = D_{r,c+1}$ on diagonal $i' = i+1$, meaning a space is inserted in the pattern at position r;
- the path is extended one cell down to cell $D_{r',c'} = D_{r+1,c}$ on diagonal $i' = i-1$, meaning a space is inserted in the text at position c;
- the path is extended one cell along the diagonal $i' = i$ to cell $D_{r',c'} = D_{r+1,c+1}$, meaning a mismatch between t_c and p_r.

Secondly, after extending a $(d-1)$-path to cell (r', c') on diagonal i', the path is further extended l cells along the diagonal i' where l is the length of the longest common prefix of $T_{c'}$ and $P_{r'}$.

As we have remarked, d-paths starting in column zero — that is, cell $(d, 0)$ — need special treatment during the initialization steps.

3.3 The algorithm

The Landau-Vishkin algorithm, presented in table 1, iterates over every diagonal i of the dynamic programming table, building d-paths that are farthest-reaching on each diagonal, beginning with all 0-paths, and then from those all 1-paths and so forth until every k-paths have been found. Those k'-paths (where $0 \leq k' \leq k$) that reach row m of the dynamic programming table are matches of P in T with at most k differences. We have omitted the special treatment of d-paths starting at column zero in for simplicity (see algorithm 1).

For each iteration we find the farthest reaching d-path on diagonal i in the following way.

- If $d = 0$ then the 0-path that starts at diagonal i is the farthest-reaching 0-path on i.
- If $d > 0$, we can find the farthest reaching d-path from the farthest-reaching $(d-1)$-paths on diagonals $i-1$, i and $i+1$ as follows.
  - We extend the farthest reaching $(d-1)$-path on diagonal $i-1$ one cell to the right to diagonal i and then further extend the path along diagonal i as described in section 3.2. In a similar way we extend the farthest reaching $(d-1)$-path on diagonal $i+1$ one cell down, and on diagonal i one cell along the diagonal i.
  - The farthest reaching d-path on diagonal i is chosen from the three paths above as being the one that has the greater index of its ending column.

In its pre-processing phase the Landau-Vishkin algorithm builds a generalized suffix tree $\mathcal{T}$ (see section 3.4) for P and T, which means that every suffix of T and every suffix of P has a corresponding leaf in $\mathcal{T}$. The tree is further pre-processed to allow for $O(1)$ lowest common ancestor (LCA) queries (see section 3.5), which enables us to find the longest common extension (LCE) of any two suffixes of P and T in constant time.

Since the body of the algorithm's inner loop runs in constant time and the pre-processing function runs in linear time, then the whole algorithm runs in time $O(kn)$, which is better than $O(mn)$ for the standard dynamic programming algorithm for large enough values of m (length of the pattern).

3.4 Suffix Trees

A suffix tree $\mathcal{T}$ for a string $T = t_1...t_n$ over an alphabet Σ is a rooted tree that has the following properties:

- there are exactly n leaves, numbered 1 to n;
- every internal node of the suffix tree, except for the root, has at least two outgoing edges;
- every edge of $\mathcal{T}$ is labeled by a substring of T, so that any two labels of all edges that start at a node v differ at least in their first characters;
- for every leaf i of $\mathcal{T}$, the concatenation of the labels of the edges on the path from the root to i gives us the suffix T_i of T.

Algorithm 1 Landau and Vishkin approximate string matching algorithm

```
1. Build a generalized suffix tree T for P and T.
2. Pre-process T so that we can answer LCA queries in constant time.
3. For every diagonal i of the dynamic programming table, find
   its farthest reaching 0-path with an O(1) LCA lookup between P and
   the (i+1)-th suffix of T.
4. For d = 1 to k:
   4.1 For every diagonal i of the dynamic programming table:
      4.1.1 Extend the farthest reaching (d-1)-path on diagonal i-1
            one cell to the right so that it reaches diagonal i
            on cell (r, s).
      4.1.2 Further extend it along i by a number of cells equal
            to the depth of the LCA of the corresponding suffixes
            of P and T (that is, P[r]...P[m] and T[s]..T[n]).
      4.1.3 Extend the (d-1)-paths on diagonals i+1 and i in a
            similar way, as described in the text.
      4.1.4 Choose the farthest reaching among the three extended
            paths.
5. Every path that reaches row m is a match of P in T with at most
   k mismatches and spaces.
```

A sentinel character which does not belong to Σ (in this paper we use both \$ and #, \$ $\neq$ #) is concatenated to T to guarantee that its suffix tree has exactly n leaves as described above. A generalized suffix tree for the strings A and B is the suffix tree for the string $A\#B$.

The suffix tree for a string of length n can be constructed in time $O(n)$ using $O(n)$ space as shown by McCreight [14] and Ukkonnen [17]. Ukkonen's algorithm has the additional property that it is *on-line*. In fact, it builds the suffix tree incrementally from the prefixes of T, starting with the smallest non-empty prefix and proceeding until the tree for the complete string is built.

Given the suffix tree $\mathcal{T}$ for the string $T = t_1...t_n$, querying if a pattern $P = p_1...p_m$ matches a substring of T takes at most $O(m)$ comparisons, independent of the length n of T.

Kurtz [11] remarks that although a suffix tree can be built using linear space complexity, most implementations actually have a large constant multiplying factor. For large enough strings, such space usage can be a limiting factor in the choice of this data structure.

3.5 Lowest Common Ancestor

The Landau-Vishkin algorithm uses a lowest common ancestor computation for finding the longest common extension of suffixes of the pattern and the text.

Definition 3 (Ancestor node) *Given two nodes v and w of a rooted tree $\mathcal{T}$, we say that a node v is an* ancestor *of a node w if v is on the unique path from the root to w. A node v is an ancestor of itself. We say that v is a* proper ancestor *of w if v is not w.*

Definition 4 (Lowest common ancestor) *In a rooted tree $\mathcal{T}$, the lowest common ancestor (LCA) of two nodes x and y is the deepest node in $\mathcal{T}$ that is an ancestor of both x and y.*

In a suffix tree $\mathcal{T}$ for the string T, given any two leaves i and j of $\mathcal{T}$, corresponding to the suffixes T_i and T_j of T, the LCA of i and j gives us the longest common prefix of T_i and T_j.

Gusfield [5] describes in detail a constant-time LCA query over a suffix tree after linear time preprocessing. It is based on a much simpler LCA algorithm for complete binary trees. The pre-processing of $\mathcal{T}$ creates an implicit mapping to a complete binary tree $\mathcal{B}$ and allows queries for lowest common ancestors of any two nodes of $\mathcal{T}$ to be answered in $O(1)$. The pre-processing also uses $O(n)$ extra space.

Farach-Colton and Bender [3] also give an algorithm for $O(1)$ LCA queries over a tree after linear preprocessing by reducing the LCA query to a *range minimum query* over an array of integer numbers that has the special property that the numbers of consecutive positions differ by exactly 1.

4 Modification of the Landau-Vishkin Algorithm

Our proposal is to substitute the use of suffix trees on Landau and Viskin's algorithm with the use of suffix arrays for computing the longest common prefixes. The advantage of this modification is that it will enable us to use a more compact data structures than even optimized implementations of suffix trees.

4.1 Suffix Arrays

Definition 5 (Suffix array) *A suffix array Pos for a string T is an array which gives us a lexicographically ordered sequence of suffixes of T.*

For the construction of the suffix array Pos, the alphabet Σ must be ordered. The sentinel character \$ is commonly used to assure the proper sorting order and it has the special property that it is either greater or smaller than any symbol of Σ.

Since it is an array of indexes of positions in T, a suffix array uses space $O(n)$, as the suffix tree, but the multiplying factor of n for the actual size is much smaller than on suffix trees ($4n$ if indexes of 32-bits are used, $8n$ if 64-bits are used). A suffix array for T can be built in $O(n)$ time from the suffix tree of T, but construction uses up too much space. An algorithm which constructs suffix arrays without an intervening suffix tree is called a *direct construction* algorithm.

An $O(n)$ space and $O(n)$ expected running time and $O(n \log_2 n)$ worst-case running time direct suffix array construction algorithm is presented in [13]. Recently $O(n)$ worst-case direct construction algorithms have been published in [10], [7], and [9].

An *enhanced suffix array* is a suffix array augmented with a table of longest common prefixes, also called an LCP table [1] [13] [7]. Given the suffix array Pos for the string $T = t_1...t_n$, the LCP table is the array lcp of n elements where $lcp[i]$ is the length of the longest common prefix of $Pos[i]$ and $Pos[i+1]$. The lcp array can be constructed in linear time from the suffix array [8], or its construction can be interleaved with the construction of the suffix array itself [9].

Many operations that we can do with suffix trees can be done with suffix arrays with the multiplication factor of $O(\log_2 n)$. The use of the LCP table can lower such factor to an addition of $O(\log_2 n)$ instead of a multiplication. Properly designed algorithms for suffix trees can be modified to run with the same bounds as a suffix tree on an enhanced suffix array as shown in [1].

4.2 Longest Common Extension Computation on a Suffix Array

What enables the Landau-Vishkin algorithm to achieve its $O(kn)$ space and running time bounds is the constant time longest common extension computation, which is done using a $O(1)$ LCA computation over a generalized suffix tree.

Given an enhanced suffix array Pos for the string $P\#T\$$, we can pre-process its corresponding lcp array and answer longest common extension queries in constant time. The key to such an operation is the following theorem.

Theorem 1 *The longest common extension between two suffixes S_a and S_b of S can be obtained from the lcp array in the following way.*

Let i be the rank of S_a among the suffixes of S (that is, $Pos[i] = a$). Let j be the rank of S_b among the suffixes of S. Without loss of generality, we assume that $i < j$. Then the longest common extension of S_a and S_b is $lcp(i,j) = min_{i \leq k < j} lcp[k]$.

Proof Let $S_a = s_a...s_{a+c}...s_n$ and $S_b = s_b...s_{b+c}...s_n$, and let c be the longest common extension of S_a and S_b (i.e. $s_a...s_{a+c-1} = s_n...s_{b+c-1}$). We assume that the string S has a sentinel character so that no suffix of S is a prefix of any other suffix of S but itself.

If $i = j - 1$ then $k = i$ and $lcp[i] = c$ is the longest common extension of S_a and S_b and we are done.

If $i < j - 1$ then select k such $lcp[k]$ is the minimum value in the interval $[i, j]$ of the lcp array. We then have two possible cases:

- If $c < lcp[k]$ we have a contradiction because $s_a \dots s_{a+lcp[k]-1} = s_b \dots s_{b+lcp[k]-1}$ by the definition of the LCP table, and the fact that the entries of lcp correspond to sorted suffixes of S.

- if $c > lcp[k]$, let $j = Pos[k]$, so that S_j is the suffix associated with position k. S_k is such that $s_j \dots s_{j+lcp[k]-1} = s_a \dots s_{a+lcp[k]-1}$ and $s_j \dots s_{j+lcp[k]-1} = s_b \dots s_{b+lcp[k]-1}$, but since $s_a \dots s_{a+c-1} = s_b \dots s_{b+c-1}$ we have that the *lcp* array should be wrongly sorted which is a contradiction.

Therefore we have $c = lcp[k]$

Thus we have reduced our longest common extension query to a *range minimum query* (RMQ) over a range in *lcp*. As it turns out, it is possible to pre-process an array of integers (such as *lcp*) in linear time so that a query for the minimum value in a given interval of the array is answered in constant-time. Kärkkäinen and Sanders give such an algorithm in [7], as does Farach-Colton and Bender in [3] and Kim et ali. in [9].

The idea presented below follows the algorithm based on Cartesian trees given in [4] by Gabow, Bentley and Tarjan.

We will then build in $O(n)$ a Cartesian tree for the *lcp* array, which will enable us to query the minimum value of any range in *lcp* in constant time by doing a constant-time lowest common ancestor query.

Definition 6 (Cartesian Trees) *A Cartesian tree for a sequence of real numbers $x_1 \dots x_n$ is a binary tree with nodes labeled by those numbers, such that the root is labeled by m where $x_m = \min x_i \mid 1 \leq i \leq n$, the left subtree is the Cartesian tree for $x_1 ... x_{m-1}$ and its right subtree is the Cartesian tree for $x_{m+1} \dots x_n$.*

Proposition 1 *The smallest number of the interval $x_i \dots x_j$ can then be found by simply finding the lowest common ancestors of nodes i and j on the Cartesian tree.*

Proof Given the nodes i and j, and v the lowest common ancestor of i and j, and suppose that $i < j$. The structure of the Cartesian tree as given above is such that if a node v is the lowest common ancestor of nodes i and j, then it means that $i \leq v \leq j$, because from the construction of the Cartesian tree, every other ancestor of v is either to the right of i and j, or to the left of them. Furthermore, from the construction of the tree, the node v is the node such that x_v is the smallest value from its subtree, and thus finding $v = LCA(i, j)$ we also find the smallest value x_v in the range $x_i \dots x_j$.

As we have remarked in section 3.5, given a tree with n nodes we can pre-process it in linear time and space so that we can answer lowest common ancestor queries over it in constant time.

We must also show how to compute the Cartesian tree in $O(n)$. It is done using the algorithm given in [4] and [3].

We build the Cartesian tree $\mathcal{C}_i$ for the array $a_1, \dots, a_i$ from the Cartesian tree $\mathcal{C}_{i-1}$ for the array $a_1, \dots, a_{i-1}$ in the following way.

- if $i = 1$, then $\mathcal{C}_1$ is a tree with a single node 1 which is labeled by a_1.
- if $i > 1$, then we build $\mathcal{C}_1$ by following the rightmost path of the tree from its leaf to the root, until we find node k such that $a_i < a_k$.
- Having found node k, We then set the right subtree of node k to be the left subtree of node i, and set node i to be the right subtree of node k.

Since each node can be added to the rightmost path at most once, and leave it at most once, the algorithm runs in time $O(n)$.

Thus, in order to answer longest common extension queries in $O(1)$ with $O(n)$ pre-processing, we do as follows.

- Build a suffix array in $O(n)$ time and space for the text concatenated with the pattern (with sentinel characters), and build the LCP table for the suffix array (either together with the construction of the suffix array itself, or in time and space $O(n)$ as given in [8]).
- Create a Cartesian tree $\mathcal{C}$ for the LCP table
- Pre-process $\mathcal{C}$ in $O(n)$ so that we can query the LCA of any two nodes of $\mathcal{C}$ in $O(1)$.

Given the suffixes i and j of the concatenated string, their longest common extension will be the result of the range-minimum-query over $lcp_i \dots lcp_j$, which is given by a $O(1)$ LCA query over the Cartesian tree $\mathcal{C}$.

4.3 Proposed Algorithm

The proposed algorithm is then the same Landau-Vishkin algorithm, substituting the suffix tree for a suffix array, and the LCA query over the suffix tree for a range-minimum query over the LCP table for the suffix array, which we give as Algorithm 2.

Algorithm 2 Modified Landau-Vishkin approximate string matching algorithm

```
1. Build a suffix array Pos for the string P#T, together with the
   lcp table
2. Build a Cartesian tree C for the lcp table
3. Process the Cartesian tree C so that we can do O(1) LCA queries
   on it.
4. For every diagonal i of the dynamic programming table, find
   its farthest reaching 0-path with an O(1) LCA lookup on the
   Cartesian Tree for the nodes corresponding to P and the i-th
   suffix of T.
5. For d = 1 to k:
   5.1 For every diagonal i of the dynamic programming table:
      5.1.1 Extend the farthest reaching (d-1)-path on diagonal
            i-1 one cell to the right so that it reaches diagonal
            i on cell (r,s).
      5.1.2 Further extend it along i by a number of cells equal
            to the RMQ of the lcp array in the range corresponding
            to the relevant suffixes of P andT (that is, P[r]...P[m]
            and T[s]..T[n]). The RMQ is given by a LCA query on the
            Cartesian tree.
      5.1.3 Extend the (d-1)-paths on diagonals i+1 and i in a
            similar way, as described in the text.
      5.1.4 Choose the farthest reaching among the three extended
            paths.
6. Every path that reached row m is a match of P in T with at most
   k mismatches and spaces.
```

Only the pre-processing phase has changed, since the RMQ is given by a LCA query, which is the same operation used in the original algorithm.

Although it also uses a LCA query over a tree, it is a smaller tree, with exactly n nodes, and which can be implemented using less space than a suffix tree.

Theorem 2 (Running time and space complexity) *The modified Landau-Vishkin approximate string matching algorithm runs in time and space $O(nk)$.*

Proof As commented before, construction and maintenance of suffix arrays run in $O(n)$ time and space ([11]) as it do building and maintenance of Cartesian trees. Since with this pre-processing LCA queries are done in $O(1)$, the construction of the approximate matches is computed in $O(kn)$ running time and space.

Although the theoretical bounds coincide with the original ones of the Landau-Vishkin algorithm, this variation uses less space during pre-processing than with suffix trees. Suffix arrays are a more compact data structure, and the LCA pre-processing for the Cartesian tree also uses less space than the same pre-processing over suffix trees, as it has exactly n nodes, while with suffix trees it ranges from $n+1$ to $2n-1$ nodes.

Supposing we are dealing with a good implementation of suffix trees where we use about 12 bytes per character (see comments on E. Coli treatment in [11]), constructed with $\frac{3}{2}n$ nodes. The space used for the lowest common ancestor pre-processing is then $12n + A\frac{3}{2}n$ bytes, where A is the space used by the LCA pre-processing for each node of the suffix tree. The suffix array and the *lcp* array can be built with a total of $8n$ bytes. If one builds the labeled Cartesian tree using $8n$ bytes (which is a reasonable assumption) such that the labels of the Cartesian tree's nodes are actual values of *lcp* (which uses $4n$ bytes), instead of indexes

to it, we can discard the suffix array altogether. Since the Cartesian tree has exactly n nodes, the final amount of space used for the preprocessing is then $12n + An$ bytes, which is an economy of $\frac{A}{2}n$ bytes over the suffix tree-based version. If a standard implementation (such as *strmat*) is used, the economy of space is much greater (see [11]). Using the LCA query algorithm presented in [5] we have $A = 12$ and the economy of space would be $6n$ bytes.

We expect the running time performance to be worse than the original algorithm, but the exact factor is yet to be determined. Even though it still has linear running time complexity, the pre-processing phase has to build the suffix array and the LCP table, and then a cartesian tree which has to be processed for $O(1)$ LCA queries, while the original algorithm only builds a suffix tree and then processes it in the same way. Since only the pre-processing phase has changed, the iteration phase runs in exactly the same time as in the original algorithm.

5 Concluding Remarks

Baeza-Yates and Gonnet [2] and Navarro [16] commented that although theoretically very interesting, in practice the Landau-Vishkin algorithm is much slower than expected. In addition, the use of suffix trees implies a large space usage, specially when combined with the necessary pre-processing for answering LCA queries in constant-time.

We have shown that it is possible to change the Landau-Vishkin approximate string matching algorithm to use enhanced suffix arrays instead of suffix trees for its computation of longest common extensions between suffixes of the text and the pattern, while keeping the same running time and space complexity. Actual space usage is slightly better than the standard algorithm because most implementations of suffix trees have a larger space usage. The impact on running time complexity is still being studied and will be made available as soon as enough data has been gathered and analysed.

References

[1] M. Abouelhoda, E. Ohlebusch, and S. Kurtz. Optimal exact string matching based on suffix arrays. In *9th International Symposium on String Processing and Information Retrieval*, pages 31–43, 2002.

[2] R. A. Baeza-Yates and G. H. Gonnet. Fast string matching with mismatches. *Information and Computation*, 108(2):187–199, 1994.

[3] M. Bender and M. Farach-Colton. The LCA Problem Revisited. In *LATIN 2000*, volume 1776 of *Lecture Notes in Computer Science*, pages 88–94. Springer Verlag, 2000.

[4] H. N. Gabow, J. L. Bentley, and R. E. Tarjan. Scaling and related techniques for geometry problems. In 16^{th} *ACM STOC*, pages 135–143, 1984.

[5] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[6] D. Hirschberg. A linear space algorithm for computing the maximal common subsequences. *Communications of the ACM*, 18:341–343, 1975.

[7] J. Kärkkäinen and P. Sanders. Simpler linear work suffix array construction. In *Int. Colloquium on Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 943–955. Springer Verlag, 2003.

[8] T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park. Linear-Time Longest-Common-Prefix Computation in Suffix Arrays and Its Applications. In *12^{th} Annual Symposium on Combinatorial Pattern Matching*, volume 2089 of *Lecture Notes in Computer Science*, pages 181–192. Springer Verlag, 2001.

[9] D. K. Kim, J. S. Sim, H. Park, and K. Park. Linear-time construction of suffix arrays. In *14th Annual Symposium on Combinatorial Pattern Matching*, volume 2676 of *Lecture Notes in Computer Science*, pages 186–199. Springer Verlag, 2003.

[10] P. Ko and S. Aluru. Space-Efficient Linear Time Construction of Suffix Arrays. *Journal of Discrete Algorithms*, to appear.

[11] S. Kurtz. Reducing the space requirement of suffix trees. *Softw., Pract. Exper.*, 29(13):1149–1171, 1999.

[12] G. Landau and U. Vishkin. Introducing Efficient Parallelism into Approximate String Matching and a new Serial Algorithm. In 18^{th} *ACM STOC*, pages 220–230, 1986.

[13] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. Technical Report TR 89-14, University of Arizona, 1989.

[14] E. M. McCreight. A Space-Economical Suffix Tree Construction Algorithm. *Journal of the Association for Computing Machinery*, 23(2):262–272, April 1976.

[15] J.I. Munro, Raman V., and Rao S. S. Space Efficient Suffix Trees. *Journal of Discrete Algorithms*, 2001.

[16] G. Navarro. A guided tour of approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[17] E. Ukkonen. On-line Construction of Suffix-Trees. *Algorithmica*, 14:249–260, 1995.

Uso de Funciones de Base Radial en Monitoreo Ambiental

Cristian Rusu
Pontificia Universidad Católica de Valparaíso, Escuela de Ingeniería Informática,
Valparaíso, Chile
cristian.rusu@ucv.cl

and

Virginia Rusu
Universitatea de Nord Baia Mare, Facultatea de Stiinte,
Baia Mare, Rumania, 4800
crusu@vtr.net

Abstract

A key problem in environmental monitoring is the surface modelling, also known as spatial interpolation. The main current approach in spatial interpolation is geostatistical. Geostatistics is neither the only nor the best spatial interpolation method. Actually there is no "best" method, universally valid. Choosing a particular method implies to make assumptions. The understanding of initial assumption, of the methods used, and the correct interpretation of the interpolation results are key elements of the spatial interpolation process.
A powerful alternative in spatial interpolation is the use of the soft computing methods. They offer the potential for a more flexible, less assumption dependent approach. Artificial Neural Networks are well suited for this kind of problems, due to their ability to handle non-linear, noisy, and inconsistent data. The present paper offers arguments for the use of Radial Basis Functions in surface modelling (spatial interpolation), based on a detailed analyze and modelling of the SIC2004 (Spatial Interpolation Comparison) dataset.

Keywords: Surface Modelling, Radial Basis Functions, Environmental Monitoring.

Resumen

Uno de los problemas básicos en el monitoreo ambiental es el modelado de superficies. El enfoque actual en la interpolación espacial es principalmente geoestadístico. La geoestadística no es el único ni el mejor método de interpolación espacial. En realidad no existe un "mejor" método, universalmente válido. La elección de un método particular de gridding implica hacer suposiciones. La correcta interpretación de las suposiciones, de los métodos utilizados y de los resultados de la interpolación son elementos claves en el proceso de interpolación espacial.
Una potente alternativa en interpolaciones espaciales es el uso de métodos de *soft computing*. Éstos son más flexibles y menos dependientes de previos suposiciones. Las redes neuronales artificiales son adecuadas para resolver problemas de interpolación espacial debido a la capacidad de manejar datos no-lineales, ruidosos, inconsistentes. El presente trabajo ofrece argumentos para el uso de funciones de base radial en modelado de superficies (interpolación espacial), en base a un detallado análisis y modelado del conjunto de datos SIC2004 (*Spatial Interpolation Comparison*).

Palabras claves: Modelado de Superficies, Funciones de Base Radial, Monitoreo Ambiental.

1. Introducción

Uno de los problemas básicos en el monitoreo ambiental es el modelado de superficies. Éste consiste de la estimación de los valores de una superficie en cualquiera de sus puntos, teniendo como datos de entrada un conjunto de datos de muestreo (x_i, y_i, z_i), generalmente con una distribución no uniforme. Los datos iniciales representan valores z medidos en puntos de coordenadas (x, y), denominados puntos de control. Los datos de forma (x_i, y_i, z_i) representan datos espaciales. El problema del modelado de superficies se conoce también como interpolación espacial, siendo parte del análisis de datos espaciales. El objetivo básico del análisis de datos espaciales consiste en obtener un modelo de un área entera partiendo de los valores z medidos en un conjunto de puntos de control. El problema del modelado de superficies ocurre en varias áreas: geología, geofísica, meteorología, ingeniería ambiental, agricultura, ingeniería, economía, medicina, sociología etc. En el monitoreo ambiental la localización de los puntos de control está definida por el posicionamiento de estaciones que forman la red de monitoreo [9], [19], [22], [23].

La precisión de las interpolaciones espaciales depende del grado de variabilidad de los valores de la superficie que debe ser aproximada. Independiente del método utilizado, las estimaciones tienen mayor precisión en las zonas de alta variabilidad y son menos precisos en las zonas de baja variabilidad. Por lo tanto es importante el estudio de las anomalías locales de la variabilidad. Las ventajas de utilizar un programa de modelación de superficies son varias:

- la posibilidad de procesamiento de una gran cantidad de datos,
- la velocidad del proceso,
- la posibilidad de comprobar varios modelos,
- la posibilidad de almacenar los modelos obtenidos, para posteriores reutilizaciones y procesamientos,
- la posibilidad de representar los resultados en varias formas gráficas y numéricas.

En el modelado de superficies se utilizan dos clases de métodos: triangulación y aproximación en red regular (*gridding*). El *gridding* genera una red (malla) regular, habitualmente cuadrática, en cual en los vértices se estiman los valores z de la superficie, en base al conjunto de puntos de control (x_i, y_i, z_i). El modelado de superficies por gridding ofrece al menos dos ventajas sobre la triangulación: (1) no es necesario conocer los puntos extremos de la superficie, y (2) los datos en red regular facilitan operaciones posteriores sobre la superficie modelada.

El enfoque actual en la interpolación espacial es principalmente geoestadístico. La interpolación geoestadística estima los valores por *kriging* (krigeado), lo cual es un interpolador exacto obtenido por la minimización de la varianza del error, tomando en cuenta la configuración espacial del fenómeno analizado.

La geoestadística no es el único ni el mejor método de interpolación espacial. En realidad no existe un "mejor" método, universalmente válido [3], [12], [19], [24]. La elección de un método particular de gridding depende básicamente de la naturaleza y las características del conjunto de datos disponibles (puntos de control) y del objetivo del modelado.

La elección de un cierto método de interpolación espacial implica hacer suposiciones. Por consecuencia, el correcto entendimiento de las suposiciones iniciales y la correcta interpretación de los resultados obtenidos basándose en estas suposiciones es una clave en el éxito del proceso. Comparaciones entre los métodos de *gridding* deben basarse en criterios como:

- los errores de aproximación de la superficie,
- la influencia de la distribución de los puntos de control,
- el volumen de datos que se debe procesar,
- la facilidad de implementar el método,
- la velocidad de cálculo.

Un enfoque alternativo para el modelado de superficies es el uso de métodos de *soft computing*. Éstos son más flexibles y menos dependientes de previos suposiciones. La inteligencia computacional, a diferencia de la computación convencional (*hard*), tolera la imprecisión, la incertidumbre, la verdad parcial, la aproximación [11].

Aunque la eficiencia de los métodos de *soft computing* en interpolaciones espaciales ha sido comprobado por varios autores ([2], [5], [8], [10], [25], entre otros), el uso de éstos en la práctica es más bien escasa. El trabajo presenta la experiencia del uso de funciones de base radial, en el marco del experimento *Spatial Interpolation Comparison* 2004 (*SIC2004*) [26].

2. Uso de Métodos *Soft Computing* en Interpolaciones Espaciales

Las redes neuronales artificiales (*Artificial Neural Networks* – ANN) son procesadores de información entrenados para identificar y expresar las relaciones implícitas, intrínsecas, de un conjunto de datos [1], [6], [7], [15], [16]. En algunos casos (como por ejemplo los datos geológicos) se requiere identificar las relaciones espaciales, mientras que en otras aplicaciones (meteorología, por ejemplo) se requiere identificar tanto las relaciones espaciales como las relaciones temporales.

Una de las aplicaciones de las ANN son las interpolaciones espaciales. Pensando en las áreas habituales que requieren análisis de datos espaciales, el problema es, fundamentalmente, uno de reconocimiento de patrones en ambiente natural, siendo por lo tanto un buen candidato para el uso de ANN. En el caso de las interpolaciones espaciales, el sistema es entrenado por un conjunto de muestras de entrenamiento, basado en un procedimiento de aprendizaje supervisado. Se considera que el sistema está entrenado correctamente si puede aproximar los valores de los patrones de entrenamiento y puede dar interpolaciones para el espacio de datos no entrenado. El problema es uno de generalización, requiriéndose una respuesta correcta a la salida para un estímulo de entrada que no ha sido entrenado con anterioridad. El sistema debe inducir la característica saliente del estímulo a la entrada y detectar la regularidad. En un enfoque de aprendizaje no supervisado, el sistema no va a utilizar muestreo de entrenamiento, sino que sólo el conjunto de datos de entrada, intentando deducir sus propiedades.

La lógica borrosa ha sido también utilizada en análisis de datos espaciales. Se utilizan los conocimientos de las especialistas, la experiencia de éstos, el análisis detallado de los datos espaciales disponibles para determinar las reglas de decisión *fuzzy*. ANN y la lógica borrosa son tecnologías que pueden complementarse con buenos resultados en el diseño de sistemas inteligentes de análisis de datos espaciales.

Las redes neuronales artificiales de tipo funciones de base radial (*Radial Basis Functions – RBF*) tienen varias aplicaciones, debido a la simplicidad, generalidad y rapidez de aprendizaje [11], [13], [14]. Las RBF son ANN unidireccionales (la información circula en un único sentido, desde las neuronas de entrada hacia las de salida), de aprendizaje híbrido (incorporan aprendizaje supervisado y no supervisado).

Habitualmente las RBF presentan una arquitectura de tres capas de neuronas:

- de entrada – envía la información de entrada hacia la capa oculta,
- oculta – compuesta por neuronas no lineales (habitualmente gaussianas),
- de salida – compuesta por neuronas lineales.

Las neuronas de la capa oculta operan en base a la distancia que separa el vector de entradas respecto del vector sináptico que cada una almacena (centroide), cantidad a la que aplican una función radial, habitualmente de forma gaussiana. Por lo tanto, estas neuronas son de respuesta localizada, respondiendo con una intensidad apreciable sólo cuando el vector de entradas y el centroide de la neurona pertenecen a una zona próxima en el espacio de entradas. Las neuronas de la capa de salida calculan la suma ponderada de las salidas que proporciona la capa oculta.

La denominación "función de base radial" destaca la simetría de estas funciones. Un nodo de la capa oculta da una salida idéntica para aquellos patrones que distan el mismo del centroide. Cuando el vector de entradas se sitúa en una región próxima al centroide de una neurona, ésta se activa, indicando que reconoce el patrón de entrada. Si el patrón de entrada es muy distinto del centroide, la respuesta tiende a cero. La propiedad expuesta recomienda el uso de RBF en el modelado de superficies (interpolaciones espaciales).

Las funciones radiales son simétricas, pero se pueden emplear también funciones asimétricas (elipsoidales), de direcciones preferenciales. Las funciones gaussianas no son el único tipo de funciones radiales que pueden utilizarse. El tipo de función radial y sus parámetros se eligen según la especificidad del problema a resolver y las características de los datos de entrada.

Algunas razones para emplear las RBF en el análisis de datos de monitoreo ambiental son las siguientes:

- el modelo ofrece una respuesta localizada y por lo tanto puede identificar las características locales de la superficie a modelar,
- las RBF son interpoladores exactos, en el sentido que la superficie modelada pasara por los puntos de control (utilizados como datos de entrada),
- el grado de irregularidad de la superficie obtenida puede controlarse a través de parámetros que puedan "suavizar" la superficie.

3. Estudio de Caso: SIC2004

Spatial Interpolation Comparison 2004 (*SIC2004*), organizado por *Joint Research Center*, bajo el patrocinio de la *Comisión Europea* consta de un ejercicio científico de interpolaciones espaciales, cuyo propósito era de comparar la eficiencia de distintos métodos de interpolación espacial, aplicadas sobre un mismo conjunto de datos de prueba. Los participantes han recibido un conjunto de *n* mediciones de una variable *X*, efectuadas en locaciones dadas, con el propósito de determinar los valores de la misma variable, en las *N-n* locaciones restantes. Los métodos/algoritmos utilizados han sido de libre elección. Al fin del ejercicio se han ofrecido a los participantes las mediciones reales efectuadas en las *N-n* locaciones restantes, para que ellos puedan evaluar la eficiencia de los métodos propuestos. El objetivo final del experimento es de apoyar el desarrollo de sistemas de decisión y advertencia para redes de monitoreo ambiental [26].

SIC2004 ha ofrecido la oportunidad de comparar los resultados obtenidos por un gran número de investigadores (34 investigadores, de 21 países), que han aplicado un amplio rango de métodos de interpolación espacial (modelado de superficies), sobre el mismo conjunto de datos. Pueden validarse así las hipótesis de trabajo, los métodos utilizados y los resultados obtenidos. Los autores del presente trabajo han participado tanto en el primer experimento SIC (SIC97, [17]), como en la edición 2004.

Los datos ofrecidos en SIC2004 son mediciones de la radioactividad natural ambiental (*natural ambient radioactivity*), realizadas en Alemania, por la red nacional de monitoreo automático (IMIS) de *German Federal Office for Radiation Protection* (BfS). Las mediciones representan valores de dosis gamma. Se ha escogido un área rectangular de aproximadamente 400 x 700 km (localizada en el Sur – Oeste de Alemania) que incluye 1008 estaciones de monitoreo. De éstas se han elegido de manera aleatoria 200 estaciones, cuyas mediciones han sido publicadas. En base a los datos publicados, los participantes del SIC2004 deberían aproximar los valores de las 808 estaciones restantes. Los valores reales medidos en las 808 estaciones han sido publicados posteriormente. La localización de las estaciones utilizadas como datos de entrada y de las estaciones cuyos valores deberían estimarse se presenta en la figura 1.

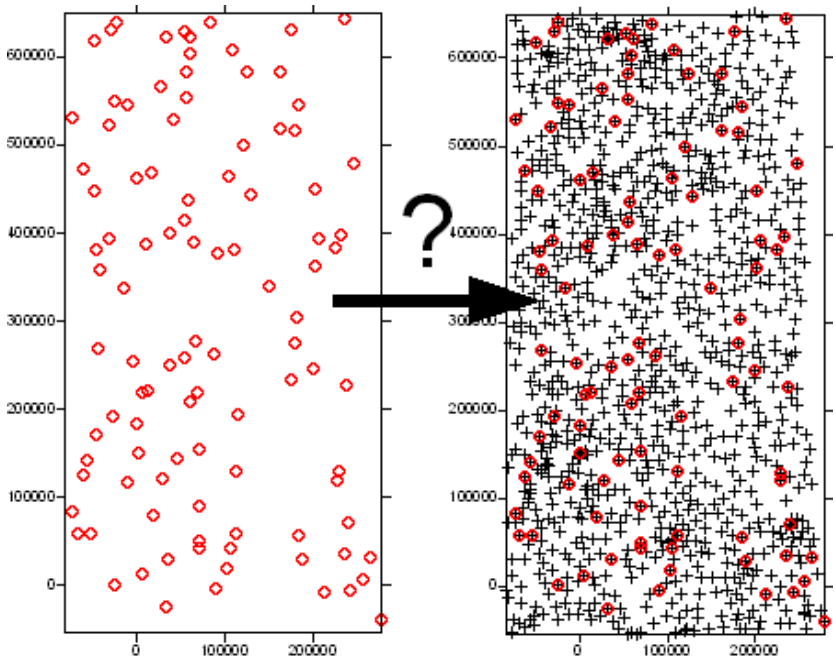


Figura 1. Localización de datos de entrada (izq.) y de datos a estimar (der.)

En el experimento SIC2004 han sido publicados, en la primera etapa:

- la posición del conjunto de datos de entrada (200 estaciones),
- la posición del conjunto de datos de salida (808 estaciones),
- 10 conjuntos de mediciones realizadas en las 200 estaciones (de entrada), en 10 días distintos.

Los 10 conjuntos de datos han sido publicados para que los participantes puedan calibrar los métodos y/o parámetros que utilizaran en el procesamiento. Posteriormente se han publicado dos conjuntos de datos a modelar, posicionadas en las mismas 200 estaciones que los conjuntos de test:

- un conjunto de mediciones reales,
- un conjunto de mediciones sobre las cuales se ha simulado una emergencia de contaminación nuclear.

Todos los conjuntos de datos han sido procesados por los autores del presente trabajo [18]. Se presentaran a continuación los resultados obtenidos sobre el conjunto de mediciones reales. El trabajo se ha enfocado en el uso de funciones de base radial (RBF). Se han utilizado arquitecturas con un número de RBF igual al número de puntos de control (200 estaciones). Se ha delimitado el área de trabajo entre las coordenadas relativas *Xmin* = -85.000, *Xmax* = 275.000, *Ymin* = -55.000, *Ymax* = 650.000. En todos los procesamientos se ha utilizado una malla regular cuadrática (*grid*), con celdas de dimensiones 5.000 x 5.000 m, de 73 columnas y 142 filas. La malla se ha dimensionado en base al número de datos de entrada y de salida, la distribución de estos, para tener una cobertura completa del área en estudio.

Los 10 conjuntos de datos de calibración se han analizado y procesado utilizándose la técnica *cross-validation*, para determinar la categoría más adecuada de funciones RBF a emplear y los parámetros de trabajo. Los mejores resultados se han obtenido con funciones de tipo multicuadrático (*multiquadratic functions*) y multilog (*multilog functions*). Se han comprobado también funciones de tipo *inverse multiquadric* y *spline*, con resultados menos satisfactorios. En base a los experimentos efectuados, se ha tomado la decisión de comprobar en detalles las funciones de tipo *multicuadrático*.

Se han analizado los datos disponibles y se ha decidido que no presentan direcciones preferenciales, por lo tanto no se ha tomado en cuanta la anisotropía. Por la misma razón, se han utilizado RBF simétricas. Se ha hecho un análisis variográfico, principalmente para determinar el rayo de influencia. Como consecuencia, se ha decidido utilizar un rayo de búsqueda de 100.000 m. En el caso de la selección preferencial de puntos de control a interpolar, se ha optado para una búsqueda por sectores (4 sectores), limitándose el número máximo de datos a utilizar en el proceso de interpolación a 10 por sector. El número de puntos de control (datos de entrada) es bastante reducido (200), lo cual no permite detectar con precisión las características locales de la superficie. Por lo tanto se ha decidido emplear una cierta suavización (*smoothing*) en el proceso de interpolación.

Los resultados obtenidos por interpolaciones espaciales se han comparado posteriormente con los valores de las mediciones reales de las 808 estaciones. Se presentan a continuación, de forma resumida, las conclusiones de dicha comparación. Se han utilizado los siguientes indicadores estadísticos:

- Error promedio (*Mean Error* - ME),
- Error promedio absoluto (*Mean Absolute Error* - MAE),
- Error porcentual (en comparación con los valores reales) promedio absoluto (PMAE),
- Error mínimo (MIN),
- Error máximo (MAX),
- Valor mínimo del error porcentual, en comparación con los valores reales (PMIN),
- Valor mínimo del error porcentual, en comparación con los valores reales (PMAX),
- Coeficiente de correlación Pearson entre los valores estimados y los valores reales (PEAR).

La tabla 1 presenta los valores de los indicadores mencionados, para distintos métodos de interpolación:

- Funciones de base radial (*Radial Basis Function* - RBF),
- Ponderación por inverso a la distancia (*Inverse Distance Weighting* - IDW),
- Geoestadística (*Kriging* - KRG),
- Aproximación local polinomial (*Local Polynomial* - LPOLY),
- Promedios móviles (*Moving Average* - MAVER),
- Curvatura mínima (*Minimum Curvature* - MCURV),
- Método Shepard (*Shepard´s Method* - SHEP).

El cuadro pone en evidencia tres métodos de mejores resultados: RBF, KRG e IDW. Cabe mencionar los resultados satisfactorios obtenidos por un método sencillo, como es IDW. Se comprueba una vez más que la elección del método de interpolación espacial influye fuertemente los resultados obtenidos. Se confirma el método RBF como el mejor, según los indicadores más significativos (MAE, PMAE, PEAR).

	ME	MAE	PMAE	MIN	MAX	PMIN	PMAX	PEAR
RBF	-1,41	9,15	9,20	-61,14	44,53	0,00	53,91	0,78
IDW	-1,36	9,93	10,11	-71,11	32,72	0,04	45,16	0,78
KRG	-1,31	9,28	9,34	-58,39	47,28	0,06	55,82	0,77
LPOLY	-1,64	10,29	10,42	-68,81	34,97	0,00	46,37	0,75
MAVER	-0,26	14,54	15,40	-81,77	33,87	0,00	59,42	0,46
MCURV	-0,80	10,36	10,47	-54,20	152,22	0,02	169,70	0,70
SHEP	-1,18	11,68	11,82	-56,67	156,50	0,03	174,47	0,64

Tabla 1. Cuadro comparativo de los indicadores obtenidos por distintos métodos de interpolación

Dentro del grupo RBF se han comparado los resultados obtenidos con distintos tipos de funciones de base radial. La tabla 2 presenta los valores de los indicadores anteriormente mencionados, para los siguientes tipos de RBF:

- multicuadrático (*multiquadratic functions* - MQ),
- multilog (*multilog functions* - MLOG),
- multicuadrático inverso (*inverse multiquadric* - INVMQ),
- spline (*natural cubic spline* – SPLINE).

	ME	MAE	PMAE	MIN	MAX	PMIN	PMAX	PEAR
MQ	-1,41	9,15	9,20	-61,14	44,53	0,00	53,91	0,78
MLOG	-1,19	9,82	9,94	-71,82	34,15	0,00	45,30	0,77
INVMQ	-12,18	199,40	190,78	-2017	1446	0,42	2004	-0,01
SPLINE	-2,94	53,23	54,59	-635,54	704,15	0,03	706,15	0,17

Tabla 2. Cuadro comparativo de los indicadores obtenidos con distintos tipos de RBF

El cuadro avala la elección de dos posibles candidatos, RBF de tipo *multicuadrático* y de tipo *multilog*, mientras que las RBF de tipo *multicuadrático inverso* y de tipo *spline* se descartan por completo. Se comprueban, por lo tanto, las conclusiones de los experimentos iniciales, sobre los conjuntos de datos de calibración (cuando se ha empleado la técnica *cross-validation*). En base a los experimentos efectuados, se ha tomado la decisión de utilizar funciones de tipo *multicuadrático*. Cabe destacar la importancia de la elección del tipo de RBF a emplear, tal como los indicadores la demuestran.

Se han probado distintos valores para el factor de suavización (*smoothing factor*). Los resultados obtenidos no varían significativamente. Puede concluirse que en el presente estudio de caso el factor de suavización no tiene mayor relevancia.

	ME	MAE	PMAE	MIN	MAX	PMIN	PMAX	PEAR
Búsqueda no referencial	-1,41	9,15	9,20	-61,14	44,53	0,00	53,91	0,78
Búsqueda preferencial	-1,28	9,30	9,37	-58,08	46,79	0,02	56,62	0,77

Tabla 3. Cuadro comparativo de los indicadores obtenidos sin y con selección preferencial de puntos de control

Se han realizado interpolaciones sin y con búsqueda (selección) preferencial de puntos de control (distinto número de sectores de búsqueda, número máximo/mínimo de puntos de control por sector, número máximo total de puntos de control a tomar en cuenta en una interpolación). Tal como se puede evaluar en la tabla 3, los indicadores obtenidos no

difieran significativamente. El cuadro presenta los indicadores obtenidos para una selección no preferencial de puntos de control y para la selección preferencial, con 4 sectores de búsqueda y un máximo de 10 puntos de control por sector.

El tiempo de cálculo requerido para las interpolaciones espaciales depende del método (algoritmo) y de los parámetros utilizados. El tiempo es levemente mayor en el caso de RBF que en el caso de métodos más sencillos, pero es comparable con el tiempo requerido por kriging. El tiempo aumenta significativamente si se utiliza una búsqueda preferencial.

3. Conclusiones

Se han comprobado en la práctica de las interpolaciones espaciales técnicas basadas en redes neuronales, específicamente funciones de base radial (RBF), sobre conjuntos de datos de radioactividad natural ambiental, disponibles en el ejercicio SIC2004. En los experimentos prácticos se han utilizado tanto RBF, como métodos geoestadísticos y métodos clásicos, comparándose los resultados obtenidos.

El estudio teórico y los experimentos prácticos realizados han comprobado la eficiencia de las RBF en interpolaciones espaciales. Las conclusiones de los experimentos iniciales, de calibración, han sido validadas por los resultados finales. El uso de RBF no presenta un grado de dificultad mayor al uso de métodos geoestadísticos. Las RBF ofrecen una alternativa muy válida al *kriging*, especialmente en el caso de conjuntos de datos de entrada (puntos de control) relativamente pequeños.

En base al estudio y a los experimentos efectuados se pueden hacer las siguientes recomendaciones metodológicas para el análisis y la interpolación de datos espaciales basados en RBF:

1) El análisis detallado de los datos espaciales, previo al modelado de superficies, es muy importante. Esto ofrecerá:
   a) indicios sobre el/los métodos de interpolación adecuados para un conjunto de dato en particular,
   b) indicios sobre los parámetros a utilizar,
   c) argumentos sobre la necesidad o la inutilidad de *clustering* sobre los datos disponibles.

2) Un método sencillo de interpolación (como IDW) es fácil de emplear y puede ofrecer indicios importantes. Es siempre recomendable aplicarlo previamente a un método más complejo.

3) La técnica *cross-validation* aplicada sobre el conjunto de datos de entrada puede ofrecer indicios muy importantes, tanto sobre las técnicas de interpolación a emplear, como sobre los parámetros adecuados.

4) La arquitectura de las RBF es determinada por el conjunto de datos mismo. Las RBF serán centradas en los puntos de control disponibles.

La importancia del tema y los resultados obtenidos incentivan posteriores experimentos. Éstos se están realizando actualmente en el marco de un nuevo proyecto de investigación [21], que pretende comprobar en la práctica otros tipos de redes neuronales artificiales y/o la combinación de éstos con otros métodos *soft computing*.

Referencias

[1] Aguilar M. y otros. Evaluación de diferentes técnicas de interpolación espacial para la generación de modelos digitales del terreno agrícola, *Mapping Interactivo*, Electronic Journal, April, 2002.

[2] Demyanov V. y otros. Neural Network Residual Kriging Application For Climatic Data, in *Mapping radioactivity in the environment*, edited by Dubois G, Malczewski J., and De Cort M., European Commission, Joint Research Center, Office for Official Publications of the European Communities, Luxembourg, 2003, pp. 176-191.

[3] Dubois G. Spatial Interpolation Comparison 97: Foreword and Introduction, *Journal of Geographic Information and Decision Analysis*, vol. 2, no. 2, 1998, pp. 1-10.

[4] Dubois G., Shibli A.R. Monitoring of environmental radioactivity: automatic mapping or expert-dependent systems?, in *Mapping radioactivity in the environment*, edited by Dubois G, Malczewski J., and De Cort M., European Commission, Joint Research Center, Office for Official Publications of the European Communities, Luxembourg,

2003, pp. 253-268.

[5] Gilardi N., Bengio S. Local Machine Learning Models for Spatial Data, *Journal of Geographic Information and Decision Analysis*, vol. 4, no. 1, 2000, pp. 11-28.

[6] Graubard S. *El nuevo debate sobre la inteligencia artificial: sistemas simbólicos y redes neuronales*, Gedisa, Barcelona, España, 1999.

[7] Hilera J., Martínez V. *Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones*, Alfaomega, Madrid, Spain, 2000.

[8] Huang Y., Wong P., Gedeon T. Spatial interpolation on overlapping partition surfaces using an optimized dynamic fuzzy-reasoning-based function estimator, in *Mapping radioactivity in the environment*, edited by Dubois G, Malczewski J., and De Cort M., European Commission, Joint Research Center, Office for Official Publications of the European Communities, Luxembourg, 2003, pp. 201-210.

[9] Jones Th. A., Hamilton, D. E., Johnson C. R. *Contouring Geological Surfaces with the Computer*, Van Nostrand Reinhold, New Zork, 1986.

[10] Lee S., Cho S., Wong P. Rainfall Prediction Using Artificial Neural Networks, *Journal of Geographic Information and Decision Analysis*, vol. 2, no. 2, 1998, pp. 253-264.

[11] Mártin B., Sanz A. *Redes Neuronales y Sistemas Difusos*, Alfaomega, Madrid, Spain, 2002.

[12] Nagy D. y otros. *Comparison of various methods of interpolation and gridding*, XX General Assembly, IUGG, Vienna, Austria, 1991.

[13] Orr M. *Introduction to Radial Basis Function Networks*, Center for Cognitive Science, University of Edinburgh, Scotland, 1996.

[14] Orr M. *Recent Advances in Radial Basis Function Networks*, Institute for Adaptative and Neural Computation, Edinburgh University, Scotland, 1999.

[15] Pao Y. *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Readings, USA, 1989.

[16] Peter A. *Retele neuronale pentru aproximarea si predictia seriilor de timp*, Presa Universitara Clujeana, Cluj-Napoca, Romania, 2000.

[17] Rusu C. Interpolarea spatiala a datelor utilizând pachetul de programe ZAZA. Studiu de caz: modelarea cantitatilor de precipitatii dupa un accident nuclear, *Lucrarile seminarului de creativitate matematica*, vol. 9 (1999-2000), Universitatea de Nord Baia Mare, Romania, pp. 135-146.

[18] Rusu C. *Modelado de superficies por métodos basados en redes neuronales artificiales*, Proyecto No. 209.736/2004, Pontificia Universidad Católica de Valparaíso, Chile, 2005.

[19] Rusu C. *Modelarea suprafetelor asistata de calculator, cu aplicatii in geologie*, PhD Thesis, Universitatea Tehnica Cluj-Napoca, Romania, September 2001.

[20] Rusu C. Neural Network Methods in Surface Modeling. Preliminary Notes., *Creative Mathematics*, Vol. 13 (2004), North University of Baia Mare, Romania, pp. 111-120.

[21] Rusu C. *Uso de métodos soft computing en el modelado de superficies*, Proyecto No. 209.737/2005, Pontificia Universidad Católica de Valparaíso, Chile, 2005.

[22] Swan A.R.H., Sandilands M. *Introduction to Geological Data Analysis*, Blackwell Science, Oxford, 1995.

[23] Watson D. *nngridr. An Implementation of Natural Neighbor Interpolation*, Vol. I of the Natural Neighbour Series, Claremont, Australia, 1994.

[24] Wingle W. Examining Common Problems Associated with Various Contouring Methods, Particularly Inverce-Distance Methods, Using Shaded Relief Surfaces, *Geotech ´92 Conference Proceedings*, Lakewood, Colorado, USA, September, 1992.

[25] Wong K.W. y otros. Rainfall Prediction Using Neural Fuzzy Technique, in *Mapping radioactivity in the environment*, edited by Dubois G, Malczewski J., and De Cort M., European Commission, Joint Research Center, Office for Official Publications of the European Communities, Luxembourg, 2003, pp. 213-221.

[26] *** www.ai-geostats.org. *The central information server for Geostatistics and Spatial Statistics*, Joint Research Center, European Commission.

Sistema experto híbrido para la determinación del factor de contratación en empresas comercializadoras de energía eléctrica

Julián Moreno Cadavid
Estudiante Maestría en Ingeniería de Sistemas, Universidad Nacional de Colombia
Medellín, Colombia
jmoreno1@ unalmed.edu.co

Demetrio Arturo Ovalle Carranza
Director Grupo de Inteligencia Artificial, Universidad Nacional de Colombia
Medellín, Colombia
dovalle@ unalmed.edu.co

Resumen

En este artículo se presenta el diseño y construcción de un sistema experto difuso para la definición de estrategias de contratación de los agentes comercializadores del mercado eléctrico Colombiano. Para lograr este objetivo se hace uso de la integración de diversas herramientas propias de la Inteligencia Artificial. Por una parte se utiliza la Lógica Difusa para modelar el conocimiento experto en la comercialización de energía eléctrica, el cual es un campo caracterizado por su alto grado de complejidad e incertidumbre; se usan las Redes Neuronales Artificiales para obtener un pronóstico de corto plazo de algunas de las variables requeridas para la inferencia; y por último se hace uso de los Agentes de Software Inteligentes para integrar el conocimiento experto del dominio, así como para brindar los mecanismos de comunicación entre los diferentes módulos que lo componen.

Palabras claves: Sistemas Inteligentes, Sistemas Expertos Difusos, Redes Neuronales Artificiales, Agentes de Software Inteligentes, Comercialización de Energía Eléctrica en Colombia.

Abstract

The aim of this paper is to present the design and building of a Fuzzy Expert System whose purpose is to determine the contract strategies used by trade agents in the Colombian Energy Market. To accomplish this goal we integrate several Artificial Intelligence tools. First of all, we use Fuzzy Logic techniques to model Expert Knowledge of Colombian energy trading, which is characterized by its high degree of complexity and uncertainty. Then, we use Artificial Neural Networks to obtain a short-term forecast of the required variables in the inference. Finally, we use Intelligent Software Agents which integrate Expert Knowledge of the Application Domain and provide communication mechanisms among several modules which compose the Multi-Agent system built into this project.

Keywords: Intelligent Systems, Fuzzy Logic Expert Systems, Artificial Neural Networks, Intelligent Software Agents, Colombian Electric Energy Market.

1. INTRODUCCIÓN

El mercado eléctrico colombiano es un ambiente altamente complejo, no solo por su alta dependencia con la hidrología que produce altas volatilidades en los precios, sino también por la incertidumbre del mercado y la regulación que lo gobierna. Por tal motivo, durante los últimos años ha surgido la necesidad del desarrollo aplicaciones inteligentes que sirvan de ayuda en la toma de decisiones a las cuales se enfrentan los comercializadores dentro de este contexto caracterizado por la competencia, y de cuyas consecuencias depende el futuro financiero de las empresas involucradas. Dichas herramientas deben servir de apoyo en este ambiente de apertura económica y liberalización de los mercados energéticos, y deben disponer de una estructura soportada en nuevos instrumentos que puedan aplicarse a dichos mercados [14]. Tales herramientas deben permitir modelar la dinámica y la interacción de todos los agentes del mercado, junto con los aspectos técnicos, económicos y de negociación aquí presentes, de manera que puedan ayudar a definir y evaluar estrategias de participación que les permitan posicionarse de una manera eficiente en la estructura actual del sector eléctrico.

La propuesta presentada en este artículo se fundamenta en algunos de los paradigmas más relevantes de la inteligencia artificial como son la Lógica Difusa, las Redes Neuronales y los Sistemas Multi-Agente. La integración de estas herramientas le proporciona a este trabajo una naturaleza híbrida, al tiempo que lo hace más robusto y escalable. Las redes neuronales artificiales son aplicadas para la predicción de las series de tiempo involucradas en los procesos de decisión, mientras que el sistema de inferencia difuso modela el conocimiento experto referente a la actividad de comercialización. Lo anterior con el fin de obtener una recomendación apropiada de cara al comercializador respecto a su proceder en la contratación de energía y/o participación en bolsa.

En el segunda sección de este artículo se presentan los antecedentes del problema a tratar, haciendo un breve recuento de la historia del sector eléctrico colombiano; en la tercera sección se describe la arquitectura del sistema, junto con la metodología utilizada para el diseño del entorno Multi-Agente; en la cuarta sección se analiza el método de predicción seleccionado para obtener el pronóstico de las variables necesarias para la recomendación; en la quinta sección se explica el método de inferencia utilizado para obtener la recomendación; en el sexta sección se presentan algunos resultados; y finalmente, en la séptima sección, se presentan las conclusiones y el trabajo futuro.

2. ANTECEDENTES

A comienzos de la década de los 90´s, un diagnóstico efectuado sobre la gestión y logros que habían alcanzado las empresas de electricidad en manos del Estado en Colombia, mostró resultados altamente desfavorables en términos de la eficiencia administrativa, operativa y financiera que dichas empresas registraban. Con este panorama, el país, a partir de la Constitución de 1991, admitió como principio clave para el logro de la eficiencia en los servicios públicos la competencia entre agentes del mercado donde fuera posible y la libre entrada a todo agente comercializador que estuviera interesado en ingresar.

En 1992, como consecuencia del severo racionamiento de energía que sufrió el país, el Gobierno expidió, haciendo uso del "estado de emergencia económica" previsto por la Constitución, el Decreto 700. Este Decreto entre otras decisiones, fijó normas para la entrada de inversionistas privados en el negocio de la generación de energía. Bajo este marco, se dio impulso a varios proyectos previstos en el Plan de Expansión y se autorizó a las empresas oficiales involucradas a firmar contratos de compraventa de energía a largo plazo con los consorcios escogidos para tales efectos. Se presentó posteriormente una reestructuración del mercado bajo las leyes 142 y 143 de 1994 (Ley de servicios públicos domiciliarios y Ley eléctrica), que tenían como finalidad aumentar la eficiencia del sistema y vincular capitales privados al mismo, y se produjo la instauración de la Bolsa de Energía en 1995 [8]. Dichas leyes definieron el marco regulatorio aplicable a las actividades de generación, así como la transmisión, distribución y comercialización de energía eléctrica; reglamentando del mismo modo los aspectos técnicos, comerciales, empresariales y operativos de estos negocios [4].

A partir de entonces los mercados de cada una de estas actividades se han tornado altamente dinámicos y complejos, obligando la adopción de estrategias eficientes que vayan de acuerdo con los marcos regulatorios vigentes por parte de las empresas que se encuentren en tal competencia [13]. En el caso particular de la comercialización, esta actividad se fundamenta en el aprovechamiento de márgenes derivados de las fluctuaciones en los precios y en las recompensas que ofrece el mercado a tomar riesgos. Es por tanto fundamental para los agentes involucrados, el grado de conocimiento que manejan y la oportunidad que éste les brinde para el análisis de la información generada por el mercado en este

nuevo esquema (CREG[1], Resolución 24 de 1995). Sin embargo, según Bedoya [2], el negocio de comercialización es quizá, desde el punto de vista estratégico, la actividad más compleja que un agente puede llevar a cabo en el mercado de energía. Los comercializadores no desempeñan funciones operativas dentro del sector, pero la complejidad de esta actividad reside en que estos agentes actúan como tomadores de riesgo de los generadores, asegurando la compra de la producción de la electricidad mediante contratos o bolsa, para luego entregarla a sus clientes o a otros agentes. Si bien esta explicación de la labor del comercializador parece simple, involucra el dinamismo que se genera en el sistema al considerar la interacción de los elementos que se presentan en los mercados de corto y largo plazo.

3. ARQUITECTURA DEL SISTEMA

El diseño y la construcción del sistema presentado se basa en uno de los más recientes logros adoptados en la ingeniería de software: el paradigma orientado a agentes. Las fortalezas de este enfoque residen en su habilidad de modelar sistemas distribuidos, en el carácter proactivo de sus componentes y en sus capacidades de comunicación [3].
La meta del sistema Multi-Agente propuesto es que las piezas de software que lo componen interactúen entre sí de forma colaborativa por medio del estándar FIPA [5]. De esta manera se logra dividir el problema en subtareas que se les asignarán a cada agente de acuerdo a sus capacidades.
De esta manera tal sociedad puede visualizarse como un sistema de procesamiento distribuido cuya arquitectura podrá depender de las condiciones del entorno como lo es la ubicación de bases de datos, la comunicación con otros sistemas, la arquitectura de la red, etc. y entre cuyas bondades están la escalabilidad, la capacidad comunicativa y el paralelismo.

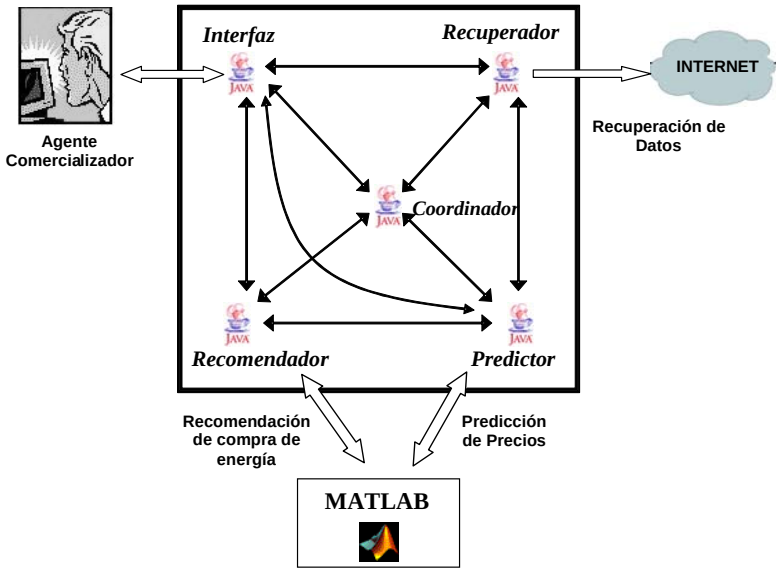


Figura 1: Arquitectura del sistema Multi-Agente

Como primera parte del sistema se identifica un modelo de inferencia que recomiende un porcentaje de compra en contratos y/o bolsa de energía para cubrir la demanda futura del comercializador, de acuerdo al comportamiento futuro del entorno. Tal proceso será llevado a cabo por un sistema de inferencia difuso que contendrá el conocimiento experto en cuanto a transacciones comerciales y que se apoyará en algunas de las variables inherentes a tal proceso como lo son el perfil del decisor, la capacidad monetaria de la empresa y otra variable exógena al usuario que es la diferencia entre el precio de bolsa y de contratos para el período correspondiente.
Así mismo, para obtener información sobre esta última variable del mercado, resulta necesario modelo de predicción para las variables precio promedio de contratos y precio promedio de bolsa, haciendo la salvedad que el horizonte del pronóstico es de corto plazo (un mes) dada la dificultad e incertidumbre que esta tarea implica.
Finalmente, y para lograr tal predicción, es necesario alimentar dicho proceso con las variables de entrada que sean necesarias, lo cual implica la creación de otro sistema que se encargue de la obtención de algunos de los indicadores más importantes del mercado, en este caso vía Web.

[1] Comisión Nacional de Regulación de Energía y Gas

De esta manera es posible concebir el sistema global como se muestra en la Figura 1. La estructura organizacional la constituye el sistema multi-agente, cuyos componentes (los agentes) tienen embebidos otros sistemas con los cuales se llevan a cabo las tareas anteriormente expuestas, además de otras como la interacción con el usuario (Interfaz) y el control de los agentes (Coordinador).

Para el diseño del sistema multi-agente planteado se usó la metodología MAS-CommonKADS [7], la cual fue desarrollada como una extensión de la metodología de ingeniería del conocimiento CommonKADS [12], agregando técnicas de metodologías orientadas a objetos. Esta metodología consta de tres fases: Conceptualización, Análisis y Diseño; en las que se proponen 7 modelos para el desarrollo de sistemas de agentes inteligentes de software, a saber:

- Modelo de Agente
- Modelo de la Organización
- Modelo de Tareas
- Modelo de la Experiencia
- Modelo de Comunicación
- Modelo de Coordinación
- Modelo de Diseño

En la fase de Conceptualización se pretende obtener una descripción general del problema que se desea resolver y elaborar un primer modelo del sistema a través de un análisis centrado en el usuario, el cual se basa en la identificación de los actores involucrados y sus respectivos casos de uso. La fase de Análisis consiste en la descripción de la arquitectura y funcionalidad del sistema multiagente a implementar, por medio de la definición los seis primeros modelos mencionados, los cuales abarcan la totalidad de los componentes del mismo así como sus relaciones. La creación de estos modelos, al igual que en las metodologías de desarrollo de software convencionales como RUP, es un proceso en espiral donde la representación de cada modelo puede alterar la especificación de los que le anteceden.

Por último, la fase de Diseño tiene como objetivo documentar todas las decisiones de diseño del sistema y determinar por una parte la arquitectura de la red de agentes, y por otra la arquitectura de agente más adecuada para cada agente. En el modelo de diseño deben considerarse los requisitos no funcionales del sistema y establecerse una relación entre los modelos de la fase de Análisis y la arquitectura del agente seleccionada en este modelo. Cada uno de estos modelos fue llevado a cabo utilizando los artefactos propios de la metodología adoptada, pero no serán expuestos en detalle en este artículo. Un análisis profundo sobre los modelos realizados en este proyecto puede encontrarse en [11].

4. MÉTODO DE INFERENCIA

Dadas las bondades de la Lógica Difusa para el manejo de la incertidumbre y con el fin de simular el comportamiento humano empleando modelos de razonamiento; se diseñó y construyó un sistema de apoyo a la toma de decisiones en la comercialización de energía basado en este paradigma buscando que fuera lo más general posible para que pudiera ajustarse a los requerimientos políticos y financieros de cualquier agente comercializador de energía.

La aplicación de modelos basados en Lógica Difusa [10] permite abordar la creación de sistemas soporte de decisión, ya que brinda la capacidad de extraer datos de forma práctica y, a través de las capacidades analíticas de los evaluadores, descubrir relaciones significativas entre ellos. Otra característica que los hace apropiados para este tipo de problemas es que estos modelos son altamente flexibles, son tolerantes a imprecisiones en los datos, pueden trabajar con funciones no lineales de diversa complejidad, y no están obligados por presunciones estadísticas acerca de las características de los datos.

El modelo de inferencia planteado consiste en un sistema de variables difusas cuyas entradas son tanto endógenas como exógenas a los comercializadores. En el primer conjunto se encuentran “la propensión a tomar riesgos en la bolsa” y “la capacidad financiera del agente”, mientras que en el segundo se encuentra uno de los indicadores del mercado mas relevantes para la comercialización como es la diferencia de precios entre las variables “precio promedio en la bolsa de energía” y “precio promedio de compra en contratos”. La salida, por su parte, corresponde a un “factor de contratación” de energía que le ayudará a tomar la decisión sobre la cantidad de demanda que debe cubrir en contratos y la que debe cubrir con la bolsa, dependiendo de las condiciones del mercado y de la política propia de la empresa.

4.1 Variables del modelo

A continuación se presenta una descripción de las variables anteriormente mencionadas, y que pueden observarse en la Figura 2 donde se presenta el diagrama que representa el sistema de inferencia.

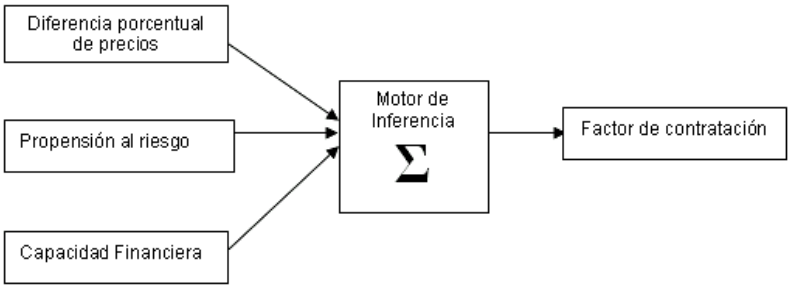


Figura 2: Estructura del sistema de inferencia difusa

- Diferencia de precios. Esta variable representa la diferencia porcentual de los pronósticos a corto plazo del precio promedio de la energía en la bolsa respecto al precio promedio de la energía en contratos para un período determinado. Su formulación se muestra a continuación:

$$Diferencia\ Porcentual\ = \frac{Pbolsa\ -\ Pcontrato}{Pbolsa}$$

En caso de ser positivo, este resultado se interpreta como el porcentaje en que el precio de la energía en la bolsa supera el precio de la energía en contratos. Por el contrario, si este resultado es negativo, la diferencia porcentual representa el porcentaje en el cual el precio de la energía en los contratos está por encima del precio en la bolsa. De tal manera, esta variable siempre debe ser interpretada respecto al precio de bolsa y así evitar ambigüedades.

- Propensión al Riesgo. Esta variable constituye la valoración en porcentaje de la disposición de los comercializadores a arriesgarse a obtener un porcentaje de la energía que requiere para cubrir sus obligaciones comerciales en la bolsa. Este factor representa las políticas comerciales que son propias de cada empresa y que reglamentan (en algunas) la toma de riesgos en la bolsa. Dichas políticas se refieren a condiciones como: Umbral mínimo de transacción en contratos, umbral máximo de transacción en bolsa para cada período, etc. Este parámetro de entrada esta definido en el intervalo [0,1]. Significando 1, la total disposición a comprar la energía en la bolsa y 0 que la totalidad de su demanda será satisfecha en contratos, independiente del comportamiento de la variable diferencia de precios.

- Capacidad Financiera. La capacidad financiera del comercializador tiene un papel muy importante a la hora de entrar en el "juego" de la compra de energía, pues ésta se traduce en un respaldo ante una eventual pérdida, es decir, que esta variable se convierte en un factor determinante a la hora de arriesgarse a comprar en bolsa. La forma en que se representa numéricamente la capacidad financiera es por medio del indicador financiero de la razón corriente, es decir, por el cociente entre el activo y el pasivo corriente. Este es un indicador estático de la liquidez, debido a que se considera los valores del activo y pasivo corriente al final del período contable, cifras que muchas veces están afectadas por aspectos circunstanciales que no reflejan la verdadera dinámica de operación de la empresa. La liquidez se refiere entonces a la capacidad del comercializador de generar fondos para el pago de sus obligaciones; es decir, la tenencia de un buen respaldo para cubrir sus necesidades económicas en cualquier momento.

- Factor de contratación. Esta variable representa la salida del sistema de inferencia difusa. La recomendación se representa en este parámetro, el cual indica al comercializador el porcentaje de su demanda total que debe tranzar en contratos para cumplir con sus obligaciones comerciales en determinado período. El valor que puede tomar este parámetro esta entre 0 y 200%. Luego, las cantidades de energía que compra en la bolsa y en contratos con agentes generadores u otros comercializadotes están dadas por:

$$Energía\ en\ contratos = D * F$$

$$Energía\ en\ bolsa = \begin{cases} D(1-F)\ sí\ F <= 1 \\ 0\ sí\ F > 1 \end{cases}$$

Siendo *D* la demanda propia y *F* el Factor de contratación

La razón por la cual se modeló esta variable hasta en un 200% es por que si el comportamiento de las variables del sistema pueden dar certeza de que se comprará mas barato en contratos que en bolsa y dependiendo de las demás variables de entrada, se podría comprar un excedente de la demanda del comercializador, que se vendería en la bolsa y obtener así ganancias adicionales.

4.2 Conjuntos difusos y funciones de pertenencia

El sistema de inferencia difusa seleccionado es de tipo Mamdani o lingüístico. Con este tipo de sistema es necesaria la construcción manual de los conjuntos difusos y de las funciones de pertenencia, los cuales se modelan a partir del conocimiento tácito de un experto y del análisis de los registros históricos de las variables involucradas.

Las funciones de pertenencia diseñadas son de forma triangular (Λ) y de forma trapezoidal (π) porque proporcionaban simplicidad en el diseño, se ajustan fácilmente para representar el conocimiento de los expertos y los tiempos de respuesta son mas rápidos en el proceso de inferencia; siendo este último factor crucial en el sistema al operar de manera distribuida.

En las Figuras 3, 4 y 5 se pueden observar el rango y las funciones de pertenencia de las variables de entrada, mientras que en el Figura 6 se presentan las de la variable de salida.

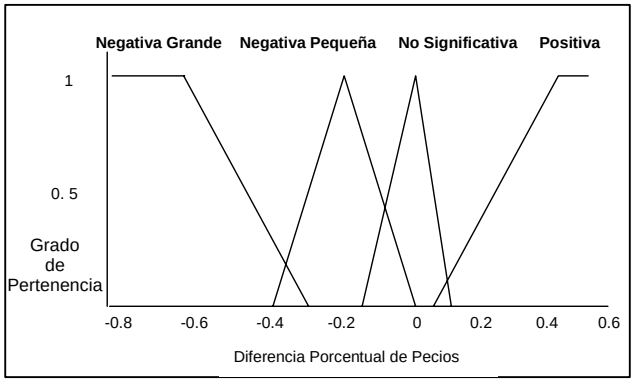


Figura 3: Conjuntos de pertenencia de la variable Diferencia Porcentual de Precios

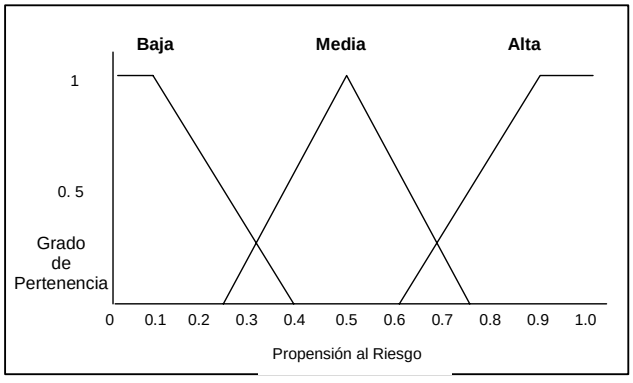


Figura 4: Conjuntos de pertenencia de la variable Propensión al Riesgo

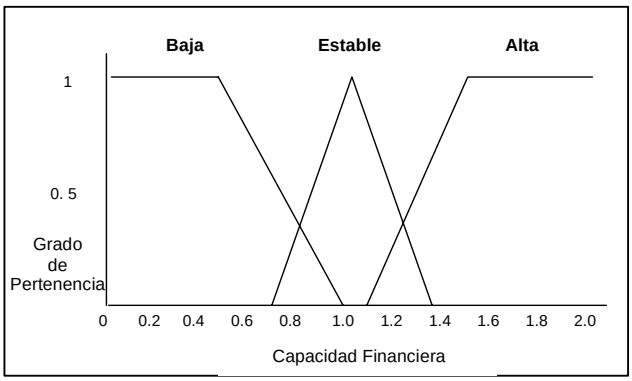


Figura 5: Conjuntos de pertenencia de la variable Capacidad Financiera

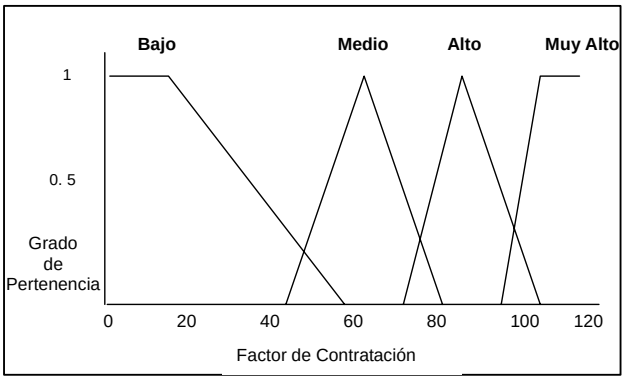


Figura 6: Conjuntos de pertenencia de la variable Factor de Contratación

En el caso de la variable Diferencia Porcentual de Precios, por ejemplo, los conjuntos de pertenencia representan en gran medida la seguridad con la cual se puede recomendar la compra en bolsa o en contratos para cumplir con las obligaciones comerciales; ya que sí por ejemplo, la diferencia porcentual es no significativa, se concluye que el riesgo de comprar en bolsa es alto. En cambio, si la diferencia porcentual es positiva y grande, se podría recomendar con bastante seguridad la compra en la bolsa.

4.3 Sistema de reglas

El conjunto de reglas que asocia cada tupla de las posibles combinaciones de las variables de entrada con la variable de salida se construye a partir de conocimiento experto. Estas reglas son del tipo:

SI <antecendente 1> Y <antecedente 2> Y … Y <antecedente n> ENTONCES <consecuente>

Donde los antecedentes corresponden a los posibles valores de las variables de entrada, mientras que el consecuente corresponde a alguno de los valores posibles de la variable de salida. Dentro del caso a tratar, y en base a las reglas empíricas conocidas por los expertos, se construyó el sistema de reglas en base a las posibles combinaciones entre las variables de entrada y su respectivo valor del porcentaje de contratación, las cuales se muestran en la Tabla 1.

Así por ejemplo, la expresión “Un pronóstico de los precios de bolsa y de contratos que indique que la diferencia porcentual entre ellos será negativa en una magnitud considerable (1), junto con una propensión al riesgo del comercializador alta (2), y un nivel de liquidez de la empresa estable (3), producen un nivel de contratación bajo (4)”, puede modelarse como la siguiente regla difusa:

SI Diferencia Porcentual de Precios ES Negativa Grande Y Propensión al Riesgo ES Baja Y Liquidez ES Estable ENTONCES Factor de Contratación ES Alto.

		Diferencia Porcentual de Precios			
Propensión al Riesgo	**Liquidez**	*Positiva*	*No significativa*	*Negativa Pequeña*	*Negativa Grande*
Alta	*Alta*	Muy Alto	Medio	Medio	Bajo
	Estable	Muy Alto	Medio	Medio	Bajo
	Baja	Muy Alto	Alto	Medio	Medio
Media	*Alta*	Muy Alto	Alto	Medio	Bajo
	Estable	Muy Alto	Alto	Alto	Medio
	Baja	Alto	Alto	Alto	Medio
Baja	*Alta*	Alto	Muy Alto	Alto	Medio
	Estable	Alto	Muy Alto	Alto	Medio
	Baja	Alto	Alto	Alto	Alto

Tabla 1: Sistema de reglas del mecanismo de inferencia

Una vez definidos los conjuntos difusos y establecido el sistema de reglas, se determina la salida del sistema de acuerdo al siguiente procedimiento: A partir de los valores numéricos de los datos de entrada, estos son *fusificados*, es decir, son categorizados de acuerdo al conjunto de pertenencia al que correspondan y se determina sus correspondientes factores de pertenencia. Posteriormente, en base a estos valores se determina las reglas que son activadas de manera simultanea y se ponderan los resultados en base a las reglas T-Norma y S-Norma [1]. Una vez obtenido un resultado, este debe *defusificarse*, es decir, traducir a un valor numérico, para lo cual se usa alguno de los mecanismos para obtenerlo [9]. En este sistema, el método elegido para la *defusificación* es el del centroide.

5. MÉTODO DE PRONÓSTICO

El agente de software encargado de realizar esta componente tiene como objetivo obtener un pronóstico a corto plazo de los indicadores precio de bolsa y precio de contratos para obtener su diferencia y alimentar así el proceso de inferencia, teniendo en cuenta la naturaleza dinámica y hasta cierto punto caótica de las mismas. En el caso del precio de contratos, tal serie no presenta variaciones importantes si se mira a largo plazo pues presenta periodos alternados de incremento y estabilidad, mientras que en el corto plazo cuenta con cierto grado de volatilidad debido a la libertad con la que cuentan los agentes del mercado para la fijación de precios. Sin embargo, en el caso del precio de bolsa la situación es bastante más dramática pues el comportamiento de la serie es muy variante y se ve afectado más fuertemente tanto por variables endógenas como exógenas al mercado.

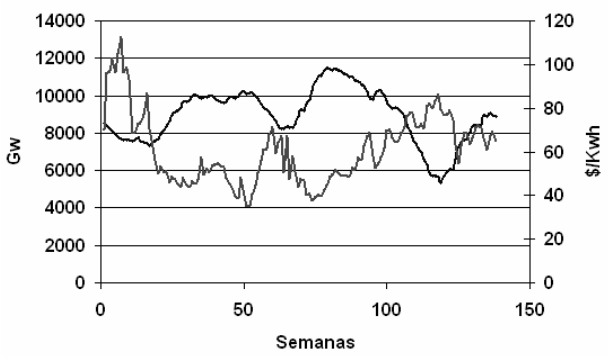


Figura 7: Precio de Bolsa versus Embalse Ofertable periodo 2000-2003

Como análisis preliminar, y a partir del registro histórico correspondiente a los últimos 4 años, se estudiaron las relaciones de estos dos indicadores consigo mismas en el tiempo, además de con otras variables que intuitivamente y por medio de la visualización de la gráfica de las series se pensó podían afectarlas. Fue así como se obtuvieron los análisis de autocorrelación y correlación cruzada con rezagos entre – 25 y 25 periodos por medio de los cuales se determinó que:

El precio de bolsa para el periodo t presenta alta autocorrelación para t-1 (R^2=0.77) y t-2 (R^2=0.56), al tiempo que muestra alta relación con el embalse ofertable en t-1 (R^2=-0.83) y t-2 (R^2=-0.61). El embalse ofertable es una medida de la capacidad de generación del sistema, con lo que estos resultados confirman la alta dependencia del sector eléctrico con la hidrología, como se muestra en la Figura 7. De manera similar ocurre con el precio de contratos en t, el cual

presenta alta autocorrelación en t-1 (R^2=0.83) y t-2 (R^2=0.64) y una relación alta con el embalse ofertable en t-1 (R^2=-0.77) y t-2 (R^2=-0.67). Finalmente, dado que se requiere el pronostico del embalse, al analizar la prueba de autoregresión respectiva se observó que el embalse ofertable en t esta altamente autorelacionado consigo mismo en t-1 (R^2=0.84) y t-2 (R^2=0.55).

De esta forma es posible visualizar el modulo de predicción como un sistema de dos niveles donde en el primero se realiza el pronóstico de las variables que dependen exclusivamente de sí mismas y cuyo resultado alimenta el segundo nivel como una de las variables de entrada, como se muestra en la Figura 8.

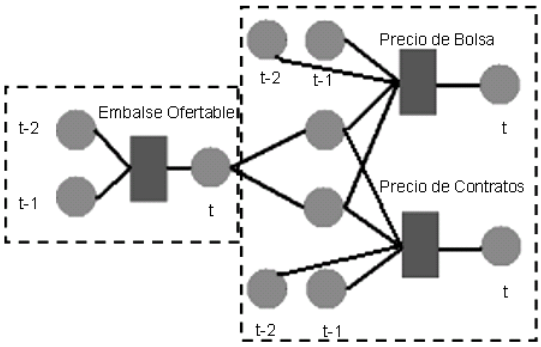


Figura 8: Estructura del sistema de predicción

Para "aprender" el comportamiento de tales series se utilizaron redes neuronales artificiales, las cuales resultan ser una excelente alternativa a los métodos lineales tanto por su robustez como por su tolerancia a fallos y su capacidad de manejar información con ruido o inconsistente. Sin entrar en demasiados detalles una red neuronal puede definirse como un procesador distribuido masivamente paralelo que cuenta con una propensión natural a reconocer y almacenar conocimiento proveniente de experiencias históricas y hacerlo disponible para su uso [6]. Para este sistema se optó por una red de 3 capas o niveles: una de entrada, una oculta y una de salida, cuya estructura se muestra en la Figura 9.

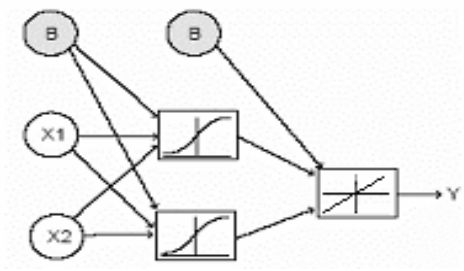


Figura 9. Arquitectura de red neuronal

En la primera capa se reciben las variables que se definieron como las entradas del sistema, en la capa oculta se realiza una agregación de las salidas de la capa anterior junto con una variable auxiliar B, que es un escalar (Bias si es igual a +1) multiplicadas por el peso de las conexiones y luego se les aplica una función de activación de modo que sus salidas pasen a la capa de salida donde se añade otra entrada auxiliar B, se agregan multiplicándolas por el peso de las conexiones correspondientes y aplicándoles finalmente otra función de activación.
Para este sistema la función de activación elegida para la capa oculta es la sigmoidea $f(x) = 1/(1 + e^{-x})$, mientras que para la capa de salida es transferencia lineal $g(x) = x$, de modo que la salida de la red neuronal puede definirse como: $Y = g(W_0 + W_j \cdot f(W_{0j} + W_{ij} \cdot X))$ donde X es el vector de entradas, W_{ij} es el vector de pesos para las conexiones de las entradas i a las neuronas j de la capa oculta, W_{0j} es el vector de pesos entre el bias y las neuronas de la capa oculta, W_j es el vector de pesos de las conexiones de las neuronas j de la capa oculta a la capa de salida, y W_0 es el peso entre el bias y la capa de salida.
Luego de establecer la salida Y, se puede determinar el error cuadrático respecto a la salida real por cada patrón de entrada como $e = (O_k - Y_k)^2$, para luego establecer el error medio cuadrático de todo el entrenamiento como

MSE = $\sum_{k=1}^{P} e_k$, siendo P el numero de patrones de entrenamiento.

El aprendizaje consiste entonces en elegir un método de optimización que de una forma iterativa y halle los vectores de pesos óptimos que minimicen tal error (RMSE), de tal forma que la salida de la red neuronal se acerque lo más posible al comportamiento de la serie estudiada. Para este efecto existen diversidad de técnicas o funciones de aprendizaje siendo las más conocidas las basadas en el gradiente descendente de la función de error, el cual en pocas palabras encuentra las derivadas parciales del error respecto a cada vector de pesos, que para la iteración 1 es un vector aleatorio en cierto rango, de tal modo que el negativo del gradiente resultante indica la dirección de máximo decrecimiento tal error, permitiéndole a la técnica de aprendizaje desplazarse en dirección de este vector con un tamaño de paso α cuya determinación depende del caso de estudio hasta llegar a un mínimo de la función.
Tal formulación puede parecer óptima para obtener un aprendizaje preciso, sin embargo, las técnicas basadas en gradiente tienen una grave falencia, y es que son absolutamente dependientes de los vectores de pesos iniciales puesto que estos determinan la posición de partida, desconociendo que el método no tiene la capacidad de salirse de mínimos locales. Por tal motivo, para este sistema se ha preferido la utilización de una técnica no tan determinística que permita "saltar" los mínimos locales en búsqueda del mínimo absoluto obteniendo resultas evidentemente mejores. Entre estos métodos se encuentran algunos como los algoritmos genéticos, el temple simulado y el quimiotactismo bacteriano de naturaleza más estocástica y entre los cuales este último ha sido el elegido por la simplicidad en su formulación y por la facilidad de su programación.

Esta técnica basada precisamente en el comportamiento de cierto grupo de bacterias para obtener alimento puede describirse como:

1. Se obtienen un vector de pesos inicial y se evalúa el MSE respectivo.
2. Al vector de pesos se le induce una perturbación ξ aleatoria.
3. Se recalcula el error, moviéndose con un tamaño de paso α en dirección del vector ξ.
4. Si el MSE presenta una disminución se agrega al vector de pesos la misma perturbación, si no se halla un ξ nuevo.
5. Se vuelve a 3 hasta que se alcance el número máximo de iteraciones o hasta que el MSE se encuentre por debajo de un límite establecido.

Adicionalmente, para mejorar la precisión de la salida y para agilizar el aprendizaje es posible usar este método solamente para hallar los pesos de la capa de entrada a la oculta, de tal forma que la determinación de los de la capa oculta a la de salida se haga por medio de mínimos cuadrados para el sistema de ecuaciones AX=b cuyas incógnitas serian los pesos de tales conexiones, los coeficientes sean dados por la salida resultante de la capa anterior y el vector b coincida con las salidas reales.

Finalmente, para asegurar que la red efectivamente aprenda el comportamiento de la serie y pueda usarse para efectos de predicción de la misma, es necesario utilizar mecanismos de validación que determinen no solo el ajuste a los patrones de entrenamiento dados si no también a un conjunto de datos de control de los cuales la red no tiene conocimiento al momento del entrenamiento. Para esto es necesario dividir los datos históricos con que se cuente en dos conjuntos llamados datos de entrenamiento y de validación. Estos últimos representan una fracción de los datos totales y son utilizados para conocer la medida del error de la red MSE ante datos desconocidos por ella pero de los cuales se conoce su salida real. Dicho conjunto de validación se construyó tomando el 10% de los datos totales en intervalos igualmente espaciados.

A partir de la evaluación del indicador del error respecto a los datos de validación de cada una de las variables estudiadas, es posible determinar que el algoritmo aquí expuesto se ajusta bastante bien a las series correspondientes, como se muestra en la Figura 10, presentando en el mejor de los casos un error promedio en el ajuste inferior a 2% y en el peor al 8%.
Sin embargo este éxito aparente no es suficiente para avalar completamente una predicción en estos casos, pues es necesario tener en cuenta que solo refleja el ajuste individual de las series, mientras que el asunto de mayor interés es determinar cual es el error inducido al alimentar la entrada de una red con la salida de otra.
Por este motivo cabe resaltar que para el desarrollo de este trabajo se optó por alimentar el modulo de inferencia únicamente con un pronóstico de corto plazo no superior a un mes, debido a la dificultad que representa un pronóstico a un mayor plazo, dada la naturaleza acumulativa del error de predicción.

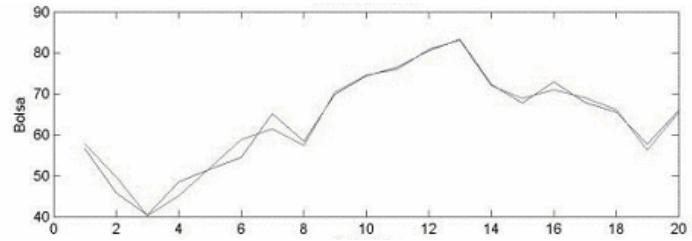


Figura 10. Resultado entrenamiento de la RN para el precio de bolsa

6. RESULTADOS

Para evaluar el desempeño del sistema se llevaron a cabo varios casos de estudio, en los cuales se ejecutó el sistema bajo diferentes condiciones y parámetros, analizando que los resultados obtenidos concordaran con el conocimiento contrastado del experto, en base a la cual se crearon las reglas de inferencia. En este análisis se tuvo en cuenta el rendimiento de cada uno de los submódulos involucrados, desde la correcta comunicación y ejecución de todos los agentes, hasta el control de los errores e intervalos de la predicción y, finalmente, la coherencia en la inferencia.

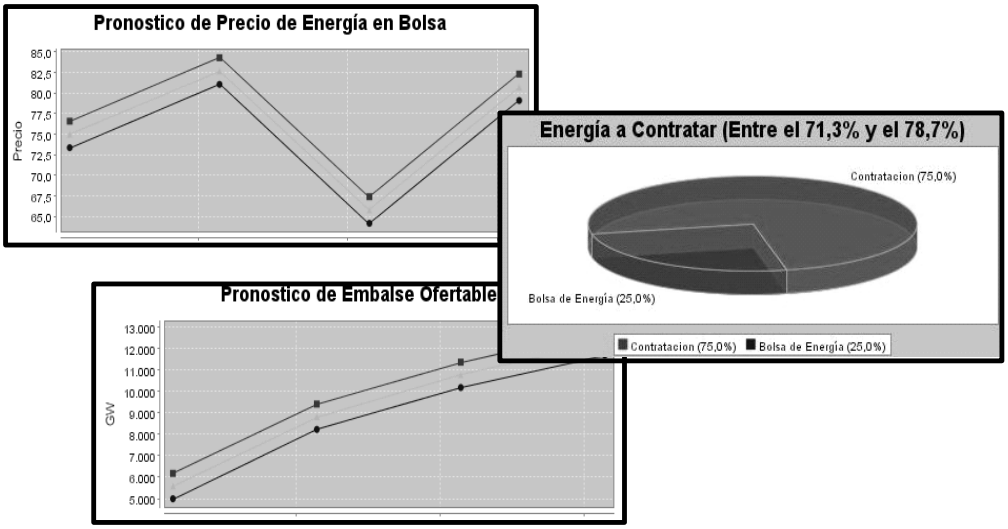


Figura 11: Resultados obtenidos del sistema

Finalmente, para analizar la eficacia del módulo de predicción, se contrastaron los datos arrojados por el sistema para los precios con los reales, obtenidos posteriormente de la base de datos NEON[2], para el periodo de análisis. Tal comparativo se presenta en la Tabla 2, donde puede observarse que, aunque las diferencias individuales de los precios reales y pronosticados superan el intervalo de predicción arrojado por la red y los valores de tales variables resultaron estar un poco por encima en ambos casos, la diferencia porcentual resultante resulta bastante consistente con la real.

Tabla 2: Comparativo datos reales vs. Pronóstico

	Real	Pronóstico	Diferencia
Precio de Contratos	74,44	76,32	-1,88
Precio de Bolsa	73,63	74,96	-1,33
Diferencia Porcentual	-0,0111	-0,0181	0,007

[2] Base de datos disponible en el MEM para los agentes del mercado registrados

7. CONCLUSIONES Y TRABAJO FUTURO

El trabajo presentado en este artículo es una primera aproximación de un sistema experto difuso para la recomendación en la compra de energía eléctrica en Colombia basado en agentes inteligentes de software que incluye procesos de recuperación de información vía Web y pronóstico de series de tiempo. Dicho sistema tiene una arquitectura basada en un sistema Multi-Agente, el cual es diseñado siguiendo las etapas de construcción de la metodología MAS-CommonKADS y hace uso dentro de los agentes de software de unos modelos de predicción e inferencia basados en lógica difusa y redes neuronales artificiales.

Sin embargo, para convertir este proyecto en una herramienta realmente robusta de tal forma que se aproxime más al comportamiento del mercado energético y se convierta en una verdadera ayuda en el quehacer diario de un agente comercializador, es necesario tener en cuenta los siguientes aspectos como trabajo futuro:

- Analizar otras variables que puedan influir de manera significativa en la decisión del comercializador y que sean menos cuantificables como puede ser el "riesgo país" o la entrada en vigencia de nuevas reglamentaciones e incluirlas en el modelo de inferencia propuesto.
- Modelar las entradas y salidas del sistema de inferencia con conjuntos de pertenencia no lineales de tal manera que la respuesta brindada por el sistema sea más exacta.
- Complementar el sistema de predicción teniendo en cuenta las holguras en las respuestas a partir del análisis del RMSE en el entrenamiento de la red, de modo que pueda realizarse la recomendación en base a las franjas donde se puede localizar el pronóstico en vez de realizarlo a partir de valores puntuales.

Reconocimientos

El trabajo presentado en este artículo fue financiado parcialmente por el proyecto de investigación titulado: "Diseño y Desarrollo de un Sistema Multi-Agente para la Simulación del Mercado Eléctrico Colombiano" auspiciado por la DIME - Dirección de Investigaciones de la Universidad Nacional de Colombia – Sede Medellín.

Referencias Bibliográficas

1] Arun D. K. Computer Vision and Fuzzy-Neural Systems. Prentice Hall. Toronto. (2001).
2] Bedoya L. Lineamientos metodológicos para el desarrollo de una herramienta de aprendizaje para la comercialización de energía eléctrica en Colombia usando Dinámica de Sistemas e Inteligencia Artificial. Tesis de Maestría en Ingeniería de Sistemas, Universidad Nacional de Colombia. (2001)
3] Davidsson, P. Multi Agent Based Simulation Beyond Social Simulation. *Proceedings of the Second International Workshop on Multi-Agent-Based Simulation*, Boston. (2000).
4] Dyner, I., Bedoya, L., Arango, S., Ochoa, P. Aprendizaje en comercialización de energía eléctrica con el apoyo de Enerbiz. *Revista de Estudios Energéticos, Facultad de Minas, Universidad Nacional de Colombia Sede Medellín*. (Diciembre, 2000).
5] García A. Introducción al estándar FIPA. *Informe técnico UCM-DSIP 98-00*. Universidad Complutense de Madrid. (2000).
6] Haykin, S. Neural Networks: A Comprehensive Foundation, Macmillan. New Cork. (1994).
7] Iglesias, C. A. Definición de una metodología para el desarrollo de Sistemas Multi-Agente, Tesis doctoral. Universidad Politécnica de Madrid. (1998).
8] ISA – MEM. Análisis del Mercado Mayorista de la Electricidad en Colombia 1998 – 1999. Gerencia de Servicios de Información, Primera Edición. (1999)
9] Kasabov, N. Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering. The MIT Press. 1998
10] Kosko, B. Pensamiento Borroso, la cueva ciencia de la lógica borrosa. Crítica. Barcelona. 1995
11] Moreno, J., Santamaría, J., Ovalle, D. Diseño y Desarrollo de un Sistema de Agentes de Software Inteligentes de Recuperación de Información, Pronóstico y Recomendación para la Compra de Energía Eléctrica. *III Congreso Colombiano y I Conferencia Andina Internacional de Investigación de Operaciones*. (2004).
12] Schreiber, Th., Wielinga, B.J., Akkermans, J.M., Van de Velde, W. CommonKADS: A comprehensive methodology for KBS development. University of Amsterdam, Netherlands Energy Research Foundation ECN and Free University of Brussels. (1994).
13] Smith, R., Montoya, S. Modelamiento de la evolución de la oferta del sector eléctrico. *En: Energética, Medellín, N° 19*. (Mayo , 1998)

4] Smith, R., Dyner, I., Franco, C.J, Bedoya, L., Arango, S., Montoya, S., Ochoa P. Microworld for Organizacional Learning in the Colombian Electricity Market, *Conferencia Internacional de Dinámica de Sistemas*, Noruega. (2000)

Sistema para detecção da borda da pupila do olho humano a partir de foto digital

Elisamara de Oliveira, MSc, PhD
Faculdade de Informática e Administração Paulista (FIAP)
Depto. de Sistemas de Informação, São Paulo, Brasil
elisa@fiap.com.br

César Augusto Cardoso Caetano, MSc, PhD
Faculdade de Informática e Administração Paulista (FIAP)
Depto. de Sistemas de Informação, São Paulo, Brasil
profcesarcaetano@fiap.com.br

Patrícia Targon Campana, MSc, PhD
Faculdade de Informática e Administração Paulista (FIAP)
Depto. de Sistemas de Informação, São Paulo, Brasil
profapatricia@fiap.com.br

Sandro Aparecido Ferraz, MSc, PhD
Faculdade de Informática e Administração Paulista (FIAP)
Depto. de Sistemas de Informação, São Paulo, Brasil
profsandro@fiap.com.br

Abstract

The presented method aims at smoothing contours of rounded medical structures detected by established techniques, but is proposed in this work as the keystone to a segmentation procedure capable to detect the center of the pupil of the human eye from digital images. The method uses as model a closed curve described in polar coordinates. The distance of the curve to its center is described by a interval-wise polinomial function. Such a curve is fitted to the data set in such a way that a measure of error is minimized. The process leads to a contour whose adherence to the data set is dictated by two main parameters: the number and size of intervals and the polynomial orders. Results have shown that the method is very powerful and extremely robust, as it is very insensitive to noise. Also, it is able to take into account both local information (details) as well as global information (general behavior) allowing a balance between data and model. It is possible to incorporate explicit geometrical knowledge into the method.

Keywords: Gaze detection, pupil contour detection, digital image processing, detection of contours.

Resumo

O método proposto tem como objetivo permitir a suavização de contornos de estruturas médicas arredondadas encontradas por técnicas já estabelecidas, mas é apresentado neste trabalho como o elemento central de um procedimento de segmentação capaz de detectar o centro da pupila do olho humano a partir de imagem digital. O método utiliza como modelo uma curva fechada descrita em coordenadas polares. A distância entre o centro dessa curva e o contorno por ela formado é descrita por uma função polinomial por intervalos. Tal contorno é ajustado de maneira ótima aos dados de entrada, que são formados por um conjunto de pontos em torno da estrutura arredondada presente na imagem. O resultado é um contorno cujo grau de aderência ao conjunto de dados pode ser controlado através do grau dos polinômios utilizados ou do número e tamanho dos intervalos. Os resultados mostram que o método é extremamente poderoso, pois é pouco sensível ao ruído presente nos dados de entrada e, ao mesmo tempo, é capaz de levar em consideração tanto a informação global (o aspecto geral do conjunto de pontos que forma os dados de entrada) quanto a informação local (os detalhes do contorno). O método permite incorporar conhecimento geométrico *a priori* sobre a classe de estruturas a ser suavizada ou detectada.

Palavras-chave: Detecção dos movimentos do olho, detecção do contorno da pupila, processamento de imagem digital, detecção de contornos.

1. INTRODUÇÃO

A técnica de *gaze detection* tem sido aplicada em várias áreas onde o movimento dos olhos é o único recurso ou o mais preciso de que se dispõe, como no caso de pesquisa médica para determinação de problemas oculares [1], caracterização de comportamento de motoristas ao dirigir [1],[6],[7], construção de interfaces direcionadas a pessoas com problemas de acessibilidade [8] (por exemplo, um tetraplégico) e sistemas de segurança [4]. Este tipo de método se baseia principalmente na estimativa do centro da pupila e, apesar de sua idéia aparentemente simples, apresenta vários problemas para uma detecção realmente eficaz do seu contorno, como altos erros na detecção em sistemas bidimensionais, a complexidade dos algoritmos utilizados em sistemas tridimensionais, a necessidade de uso de câmeras de CCD e luz infravermelha, métodos intrusivos como eletrodos e câmeras acopladas à face ou à cabeça do usuário.

O presente trabalho apresenta um método para o ajuste de estruturas arredondadas presentes em imagens médicas digitalizadas capaz de suavizar contornos descritos como um conjunto de pontos em torno da estrutura de interesse presente na imagem. O trabalho proposto neste artigo tomou como base este método, desenvolvido por Oliveira (1999) [5], e seu objetivo é fornecer uma nova técnica para a detecção do contorno e do centro da pupila do olho humano em uma imagem digital.

Esta técnica promove o ajuste otimizado do contorno da pupila aos dados de entrada, que são formados por um conjunto de pontos, contínuo ou com falhas (as falhas podem ocorrer em função da sobreposição da pálpebra sobre a pupila, por exemplo). Uma vez detectadas as bordas da pupila, o seu centro é obtido como o centro desse contorno. Testes adicionais também foram realizados com a detecção da borda da íris e o centro obtido coincide com o centro da pupila, uma vez que íris e pupila são concêntricas. O conjunto de pontos pode ser obtido a partir de métodos já estabelecidos, mas um procedimento de detecção de bordas também é proposto.

2. MÉTODO

O modelo da estrutura arredondada é descrito, em coordenadas polares, por um centro e uma função não-negativa de θ, conforme mostra a equação (1). Este modelo é denominado contorno de raio polinomial por intervalos.

$$\begin{aligned} x &= x_0 + R(\theta)\cos\theta \\ y &= y_0 + R(\theta)\,sen\,\theta \end{aligned} \qquad (1)$$

Na equação (1) x_0 e y_0 são as coordenadas do centro, $R(\theta)$ é uma função não-negativa de θ e θ é o ângulo formado entre os eixos definidos pelos pontos (x,y) e (x_0,y_0) e o eixo das abcissas, com $0 \leq \theta \leq 2\pi$, conforme ilustra a figura 1.

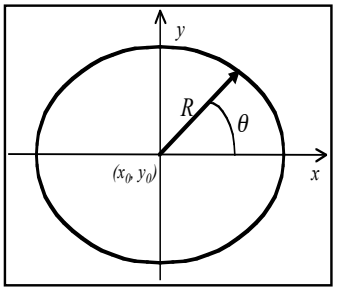


Figura 1. O modelo do contorno de raio polinomial por intervalos

O contorno que se deseja encontrar é uma curva, suave, descrita em coordenadas polares, que tenha as seguintes propriedades:

P1: A distância do centro (x_0, y_0) ao contorno, ou seja, o raio, é descrito por uma função $R(\theta)$, definindo um contorno de raios variáveis;

P2: O contorno de raios variáveis é contínuo, assim como são contínuas sua primeira e segunda derivadas;

P3: O ajuste do contorno minimiza uma função de erro, dada pelo erro quadrático entre o conjunto de pontos de entrada e o contorno de saída.

Considerando-se que o contorno ajustado esteja dividido em *N* intervalos e que os polinômios a serem utilizados sejam de grau *M*, as propriedades P1 e P2 podem ser descritas como um conjunto de equações polinomiais, cada qual válida dentro de um intervalo, levando a um conjunto de *3N* equações e (*M+1*)***N* incógnitas. O contorno ajustado será definido unicamente se, e somente se, (*M-2*)***N* equações linearmente independentes forem adicionadas às 3*N* equações definidas para o intervalo. Essas equações são obtidas usando-se a correspondência existente entre erro mínimo e derivada nula, conforme desenvolvido por Moura [2][3]. Dessa forma, chega-se a um sistema linear de ordem *N**(*M+1*), cuja solução corresponde ao contorno procurado.

Para se utilizar o método desenvolvido por Oliveira [1999] como uma ferramenta para a detecção do contorno e do centro da pupila, foi desenvolvido um procedimento que extrai suas bordas como um conjunto de pontos em torno de seu centro. Esse contorno de pontos pode ser contínuo ou conter falhas em função da pupila poder estar parcialmente obstruída em função da pálpebra estar baixa, o olho estar semi-aberto, a imagem ficar com uma marca da luz de flash da câmera digital, etc. Esse conjunto de pontos servirá de entrada para o método que, então, fará o ajuste do modelo, fornecendo um contorno suavizado da pupila e apontando o seu centro (que coincide com o centro da íris).

Para se detectar o conjunto de pontos em torno da pupila, utilizou-se um procedimento para detecção de bordas que combina limiar e magnitude do gradiente. Este procedimento é desenvolvido em duas fases. Na primeira fase aplica-se apenas um valor para o limiar. Caso as bordas resultantes da aplicação da técnica utilizando-se um único limiar não tenham conseguido detectar a região de interesse, a técnica é aplicada utilizando-se um segundo limiar. Isso só foi necesario em imganes em que a ilum não estava com o ajuste adequado.

Utilizando o conhecimento *a priori* que se tem sobre a forma anatômica de uma pupila, criou-se um "estereótipo" dela, como ilustrado na figura 2. A pupila foi dividida em quatro regiões definidas como região 1, à direita do centro da pupila, região 2, abaixo do centro da pupila, região 3, à esquerda do centro da pupila e região 4, acima do centro da pupila. O centro da íris coincide com o centro da pupila do olho humano.

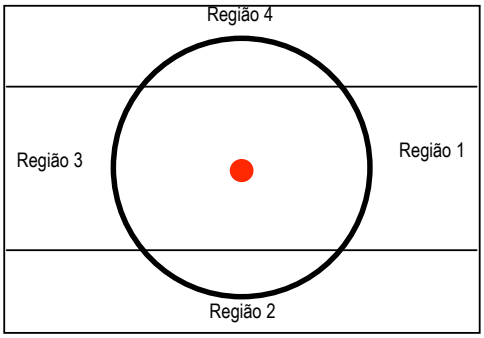


Figura 2. Modelo conceitual (estereótipo) da pupila.

Com base no estereótipo da pupila, os pontos do contorno resultantes da aplicação da técnica de detecção de bordas localizados nas regiões 1 e 3 são todos escolhidos como candidatos a pertencer ao contorno da pupila. Isso porque ao se olhar para a directa e para a esquerda, não há obstrução da pupila. Os pontos localizados nas regiões 2 e 4, são examinados um a um e cada ponto é selecionado se o valor (absoluto) da direção do gradiente neste ponto subtraído do valor do ângulo neste mesmo ponto não ultrapassar um valor determinado experimentalmente. Isso porque pode ocorrer situações em que ao se olhar para cima ou para baixo, a pupila fique parcialmente obstruída pela pálpebra. Ou seja, os pontos do contorno podem pertencer a regiões onde a informação da imagem é pouco confiável (devido à palpebra que pode interferir na imagem) ou confiável, onde não há obstruções da pupila. Deste processo resulta um conjunto de pontos com ou sem falhas, dependendo das condições da imagem original.

É importante ressaltar que a técnica de detecção de bordas e o procedimento de eliminação das bordas espúrias utilizados são bem rudimentares, fornecendo uma aproximação inicial do contorno da pupila. No entanto, a aplicação do método para obtenção do contorno final, mostra que ele é capaz de "reconstruir" o contorno da pupila, mesmo a partir de uma informação inicial parcial, o que evidencia o grande diferencial do método e sua capacidade de extrair do conjunto de pontos o modelo da estrutura procurada e encontrar seu centro de forma otimizada.

3. RESULTADOS

Uma grande potencialidade implícita no modelo de raio polinomial por intervalos é a sua capacidade de "reconstruir" contornos arredondados, a partir de um conjunto de pontos que represente esse contorno. Isso é conseguido em função da correlação entre o modelo e o conjunto de pontos que o método incorpora. No modelo utilizado pelo método, o círculo de raio polinomial por intervalos representa a geometria arredondada na qual o método se baseia para encontrar o ajuste procurado. Portanto, mesmo que o conjunto de pontos não esteja distribuído homogeneamente ao longo do contorno, o método é capaz de reconstruir a parte omitida, desde que o conjunto de pontos represente uma estrutura que se enquadre na geometria representada pelo modelo. Na figura 3 são apresentados alguns contornos sintéticos com grandes falhas em sua borda, e os resultados alcançados pelo método na obtenção do seu contorno ajustado e centro.

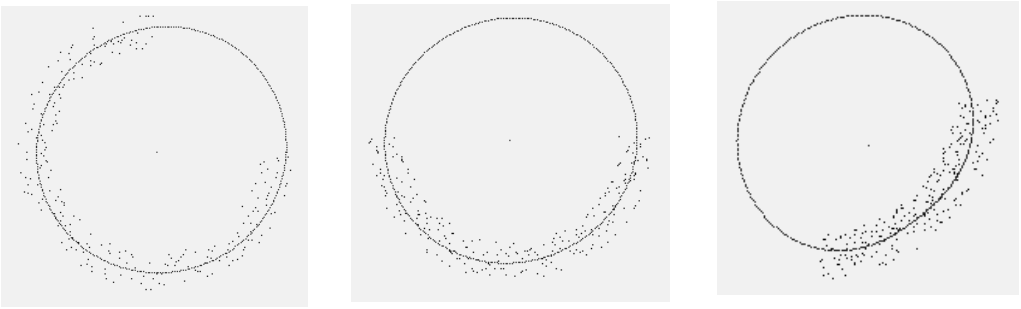


Figura 3. Contornos sintéticos com falhas e resultados (contornos e centro) obtidos pelo método

Na figura 4 são apresentadas algumas situações em que a pupila fica totalmente visível e também parcialmente oculta pela pálpebra e as suas bordas detectadas em todos os casos.

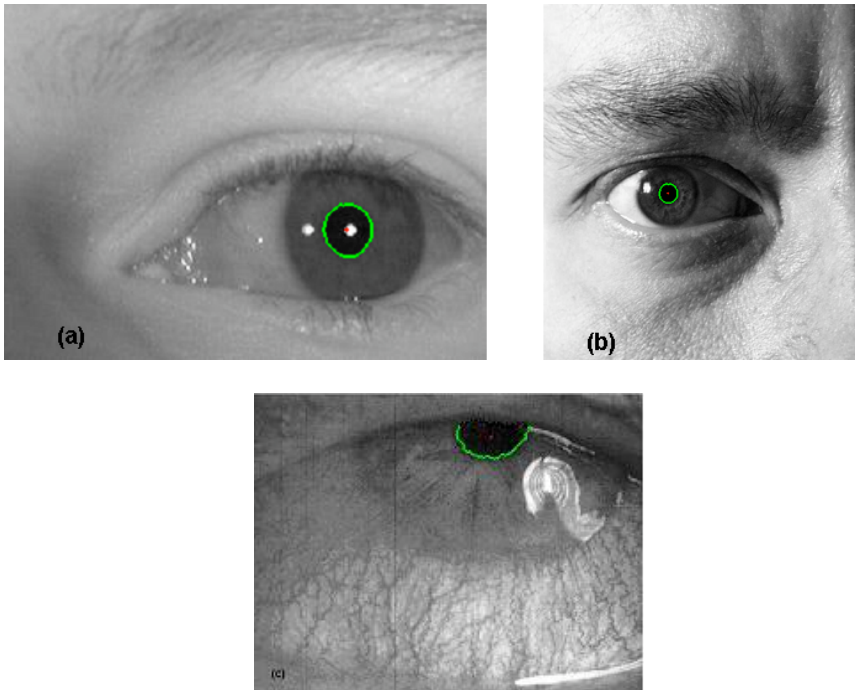


Figura 4. Exemplos de contornos obtidos pelo método de detecção de borda da pupila.

Na figura 4 podemos observar que os pontos do contorno foram selecionados a partir dos pontos inicialmente obtidos como candidatos. A aplicação do método proposto a essas imagens permite a suavização do contorno final da pupila e a obtenção de seu centro. Usando-se um valor fixo para o parâmetro *N* (número de intervalos) igual a 2, e o grau dos polinômios fixado em 3, consegue-se um ajuste em torno da pupila que pode acompanhar o conjunto de pontos e fornecer uma maior suavização, como nas figuras 4(a) e 4(b), ou reconstruir a parte oculta, como na figura 4(c).

4. CONCLUSÕES

Foi proposto um procedimento para detecção das bordas da pupila e o método foi utilizado como uma ferramenta para a suavização e/ou reconstrução dessas bordas, com a localização do centro da pupila com erro desprezível. O método mostrou-se capaz de produzir contornos suaves e fechados mesmo a partir de um conjunto de pontos com regiões desprovidas de informação. O conjunto de pontos com falhas foi o resultado de bordas obtidas de imagens nas quais a estrutura de interesse estava parcialmente encoberta pela pálpebra. Mas houve casos em que a presença de luz externa (flash, luz solar, etc.) ficou refletida na córnea do olho. As imagens utilizadas para os testes foram obtidas usando-se uma câmera digital de alta resolução e a iluminação externa foi utilizada com o objetivo de se criar melhores e piores situações de forma a simular casos críticos e casos ideais encontrados em condições reais.

Para os testes do método, procurou-se criar contornos sintéticos que fossem capazes de simular as situações mais complicadas de forma a mostrar a sua robustez. Para a aplicação em casos reais, foi desenvolvido um procedimento rudimentar de detecção de bordas da pupila que produziu contornos com ou sem falhas em torno da estrutura. Mesmo a partir dos contornos com falhas, o método foi capaz de fornecer um contorno suave, fechado e arredondado para a pupila, com a localização do seu centro. Para isso, o modelo utilizado pelo método incorporou conhecimento prévio sobre a forma anatômica típica da pupila. Os resultados comprovaram que neste tipo de aplicação reside a maior potencialidade de aplicação do método. Assim, pôde-se identificar e localizar a posição do centro da pupila, resultado que, futuramente, pode ser utilizado em um sistema que seja capaz de indicar o foco de visão de um indivíduo indicando para onde ele está olhando. Testes adicionais também foram realizados com a detecção da borda da íris e os resultados mostraram que o centro obtido coincide com o centro da pupila, uma vez que íris e pupila são concêntricas.

Em função de suas características, o método pode ser considerado uma ferramenta robusta que promove o ajuste ótimo de um modelo às formas arredondadas contidas em imagens. A área de aplicação do método é muito extensa, com interesse especial na detecção de estruturas de interesse em imagens médicas, uma vez que estruturas arredondadas estão presentes em várias áreas da Biologia. No entanto, há que se ter cautela na escolha da aplicação do método, pois, a sua tendência é de encontrar sempre um contorno arredondado, mesmo a partir de dados que formem uma geometria não arredondada. Esses casos, ao contrário de ser um defeito do método, podem ser considerados como um uso inadequado.

Referências

[1] Hortdal, JO, Jensen, PK. In-vitro measurement of corneal strain thickness and curvatura using digital image-processing. (Apr. 1993) *Acta Ophthalmologica Scandinavica*.

[2] Moura, L. A. Jr. A system for reconstruction, handling and display of 3D medical structures. London, 1988. 306p. Tese (Doutorado) – *Imperial College of Science and Technology*, University of London,

[3] Moura, L. A. Jr.; KITNEY, R. A direct method for least squares circle fitting. *Computer Physics Communications,* v. 64, n.1, (Apr. 1991) p.57-63.

[4] Matsumoto, O. Behavior Recognition Based On Head Pose and Gaze Direcrtion Measurement. *Proceeding of the 2000 IEEE/RSJ,International Conference on Intelligent Robot and systems, pp 2127- 2132, 2000.*

[5] Oliveira, E. Aplicação de contornos de raio polinomial por intervalos na detecção e suavização de contornos de estruturas arredondadas em imagens médicas. São Paulo, 1999. 168p. Tese (Doutorado) – Escola Politécnica, USP.

[6] Singh,S, and Papanikolopoulos, P. Advances in Vision-Based Detection of Driver Fatigue. *Proceedings of the ITS America Ninth Annual Meeting, 1999*.

[7] Smith, P, Shah, M, and Lobo, MV. Monitoring Head/Eye Motion for Driver Alertness with One Câmera. *Proceedings of the International Conference on Pattern Recognition, 2000.*

[8] Kim, K., and Ramakrishna, RS. Vision-Based Eye-Gaze Tracking for Human Computer Interface, Systems, Man and Cybernetics. *IEEE SMC'99 Conference Proc., No.2, pp.324-329, 1999.*

CoGrOO[1] – Um Corretor Gramatical para a língua portuguesa, acoplável ao *OpenOffice*

Jorge Kinoshita
Universidade da São Paulo (USP), Escola Politécnica
São Paulo – SP – Brasil
jorge.kinoshita@poli.usp.br

Laís do Nascimento Salvador
Univ. Cruzeiro do Sul, Centro de Ciências Exatas e Tecnológicas
São Paulo – SP – Brasil
laisns@gmail.com

Carlos E. Dantas de Menezes
Faculdades SENAC – Departamento de Ciências Exatas
São Paulo, SP – Brasil
carlos.edmenezes@sp.senac.br

Abstract

This paper describes an ongoing Portuguese Language grammar checker project, called CoGrOO -Corretor Gramatical para OpenOffice (Grammar Checker for OpenOffice). Two of its features are highlighted: - hybrid architecture, mixing rules and statistics; - free software project. This project aims to check grammatical errors such as nominal and verbal agreement, "crase" (preposition "a" + determiner "a" yielding "à"), nominal and verbal government and other common errors in Brazilian Portuguese Language. We also present some empirical results based on the implemented techniques.

Keywords: natural language processing, machine learning, statistical processing of language, free software.

Resumo

Este artigo descreve a construção em andamento de um corretor gramatical para a língua portuguesa, acoplável ao pacote de escritório OpenOffice, chamado CoGrOO (Corretor Gramatical para OpenOffice). Entre os diferenciais deste projeto, podem ser citados: - arquitetura híbrida, ou seja, alguns módulos funcionam através de estatísticas e outros são baseados em regras; - projeto de software livre. O objetivo deste projeto é verificar inadequações gramaticais tais como erros de concordância nominal e verbal, crase, regência nominal e verbal, e outros erros comuns na língua portuguesa falada no Brasil. Neste trabalho também são apresentados alguns resultados baseados nas técnicas implementadas.

Palavras chave: processamento linguagens naturais, aprendizado automático, processamento estatístico da linguagem, software livre.

1. Introdução

A ferramenta *OpenOffice* vem sendo amplamente adotada por um número crescente de usuários pessoais e corporativos. Trata-se de um pacote para escritório, composto de: processador de texto, planilha de cálculos, editor HTML e editor de apresentação. O *OpenOffice* é um projeto de código aberto e multi-plataforma, ou seja, pode ser executado sobre diferentes sistemas operacionais, como Windows, Linux, Solaris, etc. As funcionalidades do OpenOffice são comparáveis às de alguns pacotes de escritório não livres (ou proprietários) bem conhecidos, como o *Microsoft Office*. A popularização do uso desta ferramenta deve-se a dois fatores principais: o fato de ser *software* livre e sua qualidade

[1] Este projeto é patrocinado pela FINEP, Chamada Pública MCT/FINEP/Software Livre CT-INFO/FINEP-01/2003.

técnica. Mesmo com uma boa aceitação, o fato é que os usuários reclamam da falta de um módulo de correção gramatical para a língua portuguesa acoplado à ferramenta. Esse recurso já existe nos equivalentes proprietários e é de grande valia na construção de textos das mais diversas áreas de conhecimento.

O objetivo do projeto CoGrOO é construir um protótipo de corretor gramatical acoplável ao *OpenOffice*, capaz de detectar erros no texto digitado, tais como erros de concordância nominal e verbal, crase, regência nominal e verbal, e outros erros comuns na língua portuguesa falada no Brasil (por exemplo, o uso do "mal"/"mau", a confusão entre o uso de "a"/"há"), e apresentar alternativas de revisão dos mesmos.

Este artigo descreve a construção deste corretor gramatical para a língua portuguesa. Além do diferencial de ser um projeto de software livre, este projeto também apresenta contribuições nas abordagens técnicas implementadas. O texto é organizado da seguinte forma: a próxima seção fornece uma visão geral do corretor, sua arquitetura e seu funcionamento; a seção de conclusão apresenta as contribuições deste trabalho, como também uma análise comparativa com trabalhos correlatos.

2. ARQUITETURA

O sistema CoGrOO é composto por módulos responsáveis pela análise da sentença e por módulos de detecção de erros gramaticais. O primeiro módulo do corretor gramatical é o Separador de Sentenças: ele recebe o texto do *OpenOffice*, separa e armazena as sentenças contidas no texto. O segundo módulo, o Etiquetador Morfológico, recebe como entrada cada uma das sentenças e atribui etiquetas morfológicas aos seus itens lexicais (palavras e pontuações).

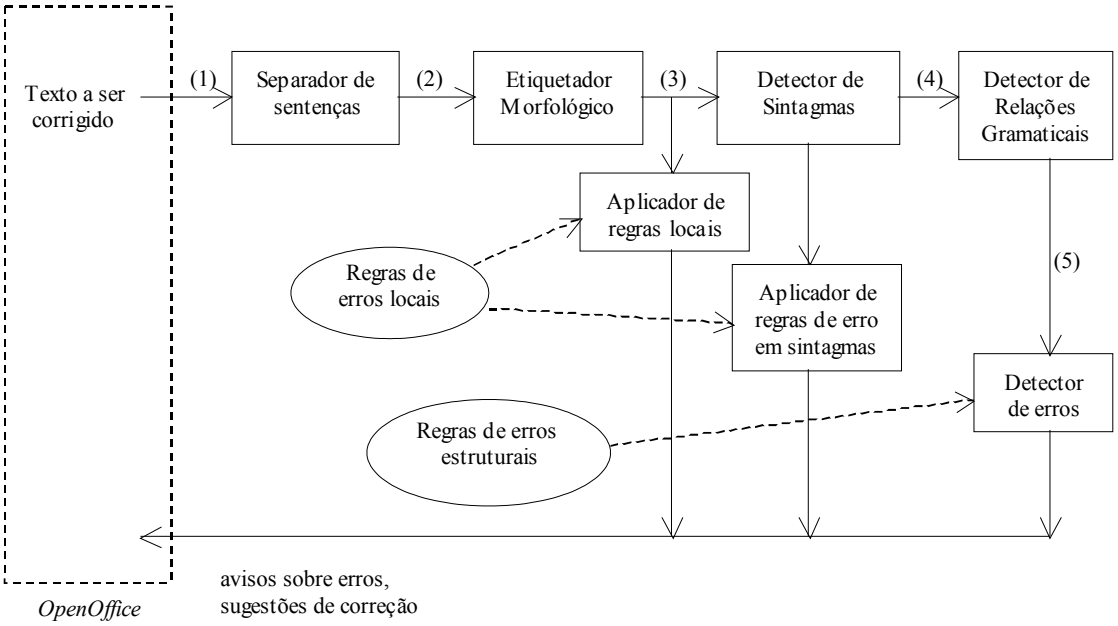


Figura 1 – Arquitetura do sistema CoGrOO

Após a classificação morfológica, submete-se o texto a outros dois módulos: o Detector de Sintagmas e o Detector de Relações Gramaticais; o primeiro agrupa seqüências de nomes/artigos/adjetivos (sintagmas nominais) ou seqüência de verbos (sintagma verbal); o segundo atribui relações gramaticais a estes agrupamentos, estabelecendo os papéis gramaticais (sujeito, verbo, predicado), quando possível, de cada agrupamento.

A arquitetura do CoGrOO com os seus principais módulos está representada na Figura 1. Como se observa nesta, existe um módulo de análise de erro para cada etapa da análise da sentença. A primeira fase do processo de correção gramatical se aplica a uma seqüência curta de palavras e/ou etiquetas morfológicas com base em regras de

erros, chamadas regras locais. Para o tratamento de erros mais complexos como erros de concordância verbal, na segunda fase é implementada uma abordagem combinada de detecção de relações gramaticais e de aplicação de regras de erros estruturais. Nas regras estruturais são usadas seqüências de palavras e/ou etiquetas gramaticais associadas ao texto analisado. O processo de revisão de erros gramaticais é acionado inicialmente após a etiquetagem morfológica e é também realizado concomitantemente com o trabalho de detecção de sintagmas e de atribuição de relações gramaticais. Nos itens seguintes, será descrito mais detalhadamente cada módulo do sistema.

2.1. Separador de sentenças

A função do separador de sentenças é preparar o texto para o etiquetador morfológico. Ele recebe como entrada o texto a ser analisado, marca o texto com sinais especiais, levando em consideração os casos de abreviação. Este módulo consulta um dicionário de abreviações da língua portuguesa e sua saída é uma estrutura de dados para cada sentença encontrada no texto.

Segue um exemplo de texto de entrada para este módulo e a respectiva saída produzida:

Entrada (1)[2]: Ele foi procurar uma casa. Ele vai se casar com a Sra. Maria.
Saída (2): Ele foi procurar uma casa . (1ª. sentença)
Ele vai se casar com a Sra. Maria . (2ª. sentença)

Pode-se observar que o ponto em "Sra." não foi confundido com um ponto final, pois se trata de uma abreviação.

2.2. Etiquetador Morfológico

O papel do etiquetador é associar uma etiqueta morfológica para cada palavra da sentença. Para a atribuição das etiquetas morfológicas, o etiquetador executa as seguintes tarefas, nesta ordem: (1) associa etiquetas possíveis às palavras da sentença; (2) avalia a probabilidade de cada etiqueta; (3) define a etiqueta mais provável para cada palavra da sentença, examinando o contexto.

A tarefa de se associar etiquetas possíveis às palavras das sentenças é realizada através do acesso aos bancos de dados léxicos (chamados "dicionários"), que são gerados pelo processamento de um *corpus* anotado [3], procedendo assim uma tarefa de aprendizado automático. Dessa forma, tenta-se associar a cada palavra uma ou mais etiquetas.

Caso não seja possível encontrar a palavra no dicionário, tenta-se descobrir a sua etiqueta através da pesquisa das últimas letras dessa palavra (sufixo) num dicionário de sufixos, também gerado pelo processamento de *corpus* anotado. Caso exista este sufixo, adota-se a etiqueta correspondente. Caso não exista, a palavra é etiquetada como substantivo singular, visto que esta é a etiqueta mais comum em nossa língua.

Através da consulta aos dicionários é possível a associação de mais de uma etiqueta à mesma palavra, de modo que torna-se necessária a aplicação de uma técnica para resolução de ambigüidades:

- monta-se um banco de dados com seqüências de três etiquetas morfológicas consecutivas (trigramas) presentes no *corpus* anotado, com suas respectivas freqüências. Por exemplo, na sentença "o gato comeu o rato" encontramos os trigramas: artigo-substantivo-verbo, substantivo-verbo-artigo etc.
- a heurística envolve levar em conta tanto a freqüência de cada etiqueta associada àquele item lexical, quanto a freqüência dos trigramas possíveis que incorporam esta etiqueta. Da combinação mais provável é escolhida a etiqueta para a classificação deste item lexical.

Para ilustrar o trabalho do etiquetador, segue a simulação para a 1ª. sentença citada na seção anterior:

Entrada (2): Ele foi procurar uma casa .

A tabela 1 mostra as etiquetas morfológicas e suas respectivas notas (obtidas pela heurística comentada) associadas a cada item lexical. Neste exemplo percebe-se que, com exceção do ponto final, sempre há necessidade de lidar-se com situações de ambigüidade.

[2] Os números entre parêntesis referem-se à Figura 1.

Palavras	**Etiquetas Morfológicas**
Ele	PERS_M_3S_NOM_(16740) PERS_M_3S_NOM/PIV_(62) N_M_S_(6)
foi	{ir}_V_PS_3S_IND_VFIN_(2753) {ser}_V_PS_3S_IND_VFIN_(55500) ADV_(138)
procurar	{procurar}_V_FUT_1/3S_SUBJ_VFIN_(1) {procurar}_V_INF_(759) {procurar}_V_INF_0/1/3S_(10) {procurar}_V_INF_3S_(33) {procurar} V_FUT_3S_SUBJ_VFIN_(15)
uma	NUM_F_S_(12037) DET_F_S_(96697) NUM_M/F_P_(32) NUM_M_P_(2) NUM_F_P_(16)
casa	{casar}_V_PR_3S_IND_VFIN_(114) N_F_S_(6034) {casar}_V_IMP_2S_VFIN_(1)
$.	-PNT_NS(1108953)

Tabela 1 – Saída do Etiquetador Morfológico (3)

A palavra “casa”, por exemplo, pode ser um substantivo feminino singular (N_F_S), um verbo no presente (V_PR) ou um verbo no imperativo (V_IMP). Através das notas associadas a essas etiquetas, escolhe-se N_F_S como a mais provável para “casa”.

2.2.1. Aplicação de regras locais

A partir do texto etiquetado morfologicamente é aplicada a primeira bateria de regras de erros gramaticais. Estas regras são denominadas locais, pois envolvem um contexto muito curto, de poucas palavras ou etiquetas à esquerda ou à direita, sem levar em conta a estrutura da sentença. Este processo de revisão gramatical aponta alguns erros comparando em tempo real a seqüência de etiquetas morfológicas (substantivo, verbo, numeral, artigo etc.) das palavras digitadas com padrões armazenados nas regras locais. Caso a seqüência apresente uma ordem incomum, o sistema procura e apresenta a ordem correta, ou a mais comum.

Exemplos de erros detectados nesta fase: uso de crase antes de palavra masculina ou de verbo, o provável uso flexionado do advérbio meio, etc.

Uma entrada no arquivo de regras locais possui 4 componentes:

- a regra de erro, isto é, uma expressão regular com palavras e etiquetas morfológicas que especifica um padrão de erro gramatical;
- a mensagem explicativa sobre o diagnóstico do erro;
- um exemplo do uso adequado do padrão gramatical;
- um exemplo do uso não adequado do padrão gramatical.

"à" N_M_S_ **mensagem:** O sinal indicativo de crase indica que temos “a + a” expressos em um só “à”. Somente ocorre crase quando há encontro de preposição a com artigo ou pronome demonstrativo a/as. **adequado:** Refiro-me ao trabalho remunerado./Refiro-me a trabalho remunerado. **inadequado:** Refiro-me à trabalho remunerado.
"(meia|meias)" _ADJ **mensagem:** A palavra “meio” usada no sentido de “um pouco” é advérbio, portanto invariável. Exceção: A palavra “meia” usada como substantivo concorda com o adjetivo. **adequado:** A conclusão está meio confusa/ As conclusões estão meio confusas. **inadequado:** A conclusão está meia confusa/ As conclusões estão meias confusas.

Tabela 2 – Exemplos de regras locais (vide Figura 1)

Na tabela 2 são apresentados dois exemplos de regras locais: a regra para detecção do uso de crase antes de palavra masculina singular, cuja etiqueta morfológica é N_M_S_; e a regra para detecção do uso do advérbio meio antes de um adjetivo, cuja etiqueta morfológica é _ADJ.

2.3. Detector de sintagmas

O objetivo deste módulo é localizar pequenos sintagmas nominais ou verbais, também chamados *noun phrase* (NP) e *verbal phrase* (VP). Um sintagma é uma unidade da análise sintática composta por um núcleo e outros termos associados, formando uma locução – conjunto de palavras que equivalem a um só vocábulo – que faz parte da estrutura da sentença. Os sintagmas se classificam de acordo com os seus elementos nucleares, neste trabalho detecta-se o sintagma nominal (NP), quando o núcleo do sintagma é um nome; e o sintagma verbal (VP), quando o núcleo do sintagma é um verbo. A implementação do detector de sintagmas é baseada numa máquina de estados que varre a sentença etiquetada morfologicamente na busca de seqüências de artigos, substantivos e adjetivos, no caso do sintagma nominal; ou seqüências de verbos e advérbios no caso dos sintagmas verbais.

Voltando ao exemplo apresentado nas seções anteriores:

Entrada (3):
Ele:[PERS_M_3S_NOM_] foi:[{ir}_V_PS_3S_IND_VFIN_] procurar:[{procurar}_V_INF_] uma:[DET_F_S_] casa: [N_F_S_] $,:[-PNT_NS]

Saída (4):

Palavras	**Etiquetas Sintáticas**
Ele	NP_3S_M_0_
foi	VP_3S_0_
procurar	
uma	NP_3S_F_1_
casa	
.	-PNT_NS

Tabela 3 – Saída do Detector de Sintagmas (4)

Esta saída (tabela 3) indica que foram detectados três sintagmas na sentença: dois nominais (NP) e um verbal (VP). Além desta saída, este módulo fornece a informação sobre a posição das palavras aglutinadas. A etiqueta de um sintagma nominal/verbal apresenta as seguintes informações adicionais:

- a pessoa associada ao sintagma – no exemplo, os três sintagmas estão na 3ª pessoa do singular (3S);
- o gênero associado ao sintagma, informação restrita aos sintagmas nominais – no exemplo, o primeiro sintagma nominal é do gênero masculino (M) e o segundo é do gênero feminino (F);
- a última parte da etiqueta (os números 0 e 1) representa uma numeração dada aos sintagmas dentro da sentença.

2.3.1. Aplicação de regras de erro em sintagmas

Na detecção de um sintagma nominal são aplicadas regras de erros locais, como por exemplo: erros de concordância entre artigo e substantivo, entre substantivo e adjetivo.

A tabela 4 apresenta dois exemplos de regras de erro aplicadas nesta fase:

N_F_S_ ADJ_F_P_ **mensagem:** O adjetivo concorda em gênero (masculino ou feminino) e número (singular ou plural) com o substantivo a que se refere. **adequado:** Faces rosadas da criança ou a face rosada da criança. **inadequado:** Face rosadas da criança.
DET_M_S_ (N_M_P_|N_F_S_|N_F_P_) **mensagem:** Os artigos definidos (o, a, os, as) concordam em número (singular ou plural) e em gênero (masculino ou feminino) com o substantivo a que se referem. **adequado:** Os copos caíram da mesa. A roupa está no guarda-roupa. As roupas estão no guarda-roupa. **inadequado:** O copos caiu da mesa. O roupa está no guarda-roupa. O roupas estão no guarda-roupa.

Tabela 4 – Exemplos de regras de erro em sintagmas (vide Figura 1)

2.4. Detector de relações gramaticais

O objetivo deste módulo é levantar relações gramaticais como sujeito-verbo, verbo-objeto, verbo-preposição, na sentença a ser analisada. Na fase atual do projeto está sendo implementada a detecção da relação sujeito-verbo. Para isso, foi implementada a seguinte estratégia: no texto analisado é detectada uma seqüência de etiquetas e sintagmas que se encaixam com um padrão que indica que um determinado sintagma nominal é sujeito de um sintagma verbal. Um exemplo de padrão sujeito-verbo é onde "!" significa começo de sentença. Nesse caso o corretor verifica a concordância verbal entre NP e VP.

Os padrões de relações sujeito-verbo foram extraídos do *corpus* anotado com informações morfo-sintáticas. Se esse padrão se repetir muitas vezes é uma indicação de uma relação gramatical válida. No *corpus* observa-se que o para o padrão : "! NP VP", NP é sujeito de VP em 85% dos casos. Devido à esta alta freqüência adota-se "! NP VP" como um padrão para se detectar sujeito nas sentenças a serem analisadas pelo corretor gramatical. Um padrão corresponde a:

<elemento> NP <elemento>* VP , ordem direta (sujeito antes do verbo)

VP <elemento>* NP <elemento> , ordem inversa

,onde <elemento> = ! ou etiqueta ou sintagma

Os padrões encontrados no *corpus* servem de base para uma máquina de estados que é usada pelo corretor gramatical na detecção de erros de concordância entre sujeito e verbo. Uma vez que um padrão foi encontrado na sentença analisada é produzida uma etiqueta sintática [SUBJ>] associada ao NP e a etiqueta sintática (M)VP associada a VP.

Voltando ao exemplo em questão:

```
(2) Ele foi procurar uma casa.
```

O detector de relações gramaticais produz a seguinte saída (5):

```
|- [ [SUBJ>]_M_3S_ ]
   |- [ (SJ>)SN_M_3S_0_ ]
      |- [ SN_M_3S_0_ ]
         |- PERS_M_3S_NOM_ --> Ele
|- [ (M)VP_3S_0_ ]
   |- {ser}_V_PS_3S_IND_VFIN_ --> foi
   |- {procurar}_V_INF_ --> procurar
|- [ SN_F_3S_1_ ]
   |- DET_F_S_ --> uma
   |- N_F_S_ --> casa
|- -PNT_ABS --> $.
```

O corretor detectou o sintagma nominal "Ele" como sujeito da sentença e produziu a etiqueta sintática [SUBJ>] _M_3S_ (sujeito masculino na 3ª. pessoa do singular). Foi também detectado o sintagma verbal "foi procurar" e foi produzida a etiqueta sintática (M)VP_3S_0_ (M, verbo principal, na 3ª. pessoa do singular, relacionado ao sujeito).

2.4.1. Processo de Correção Gramatical

Durante a fase de detecção de relações gramaticais, o processo de correção gramatical é implementado de duas formas:

- através da verificação de informações contidas nas etiquetas sintáticas;
- baseada em regras de erros estruturais.

Um tipo de erro detectado com base na inspeção das etiquetas sintáticas é o erro de concordância verbal. O processo é bastante simples: com as informações contidas nas etiquetas sintáticas, verifica-se a correspondência entre a pessoa do sujeito e a do verbo a ele associado.

No exemplo anterior tem-se uma correspondência entre a pessoa do sujeito e a do verbo, no caso, 3ª. pessoa do singular, e não é acusado erro de concordância verbal. No caso da frase "Os corretores foi procurar uma casa", será produzida a seguinte seqüência de etiquetas e o seguinte aviso de erro gramatical:

```
Sentença: Os corretores foi procurar uma casa.

|- [ [SUBJ>]_M_3P_ ]
   |- [ (SJ>)SN_M_3P_0_ ]
      |- [ SN_M_3P_0_ ]
         |- DET_M_P_ --> Os
         |- N_M_P_ --> corretores
|- [ (M)VP_3S_0_ ]
   |- {ser}_V_PS_3S_IND_VFIN_ --> foi
   |- {procurar}_V_INF_ --> procurer
|- [ SN_F_3S_1_ ]
   |- DET_F_S_ --> uma
   |- N_F_S_ --> casa
|- -PNT_ABS --> $.

Mensagem de erro: Verificou-se erro de concordância entre o verbo e o sujeito.
```

Neste exemplo, o sintagma "Os corretores" recebeu a etiqueta sintática [SUBJ>]_M_3P, que não concorda em número com a etiqueta sintática (M)VP_3S, associada ao sintagma "foi procurar".

Outros erros, que envolvem relações gramaticais, estão relacionados ao uso inadequado da língua em situações bem específicas. Nestes casos são usadas as chamadas Regras Estruturais.

Uma regra estrutural de erro é uma expressão regular que envolve palavras e etiquetas morfológicas, como no caso das regras locais, porém, também engloba informações sintáticas. Uma aplicação de regra estrutural encontra-se no tratamento de inadequações do uso do verbo "fazer" na indicação de tempo. Para a resolução desses problemas foi criada uma regra estrutural que detecta um erro em expressões do tipo "fazem 20 anos". Esta regra só é aplicada desde que a palavra "fazem" não seja um verbo associado a um sujeito na terceira pessoa do plural. Logo, numa frase deste tipo "Fazem 20 anos que não o vejo", é detectado um erro gramatical, e numa frase como "Hoje eles fazem 20 anos de casados", não é detectado erro gramatical. Dessa forma pretende-se reduzir as situações de falsos positivos, em que uma abordagem com apenas regras de erros locais pode incorrer.

3. Resultados

Embora o projeto CoGrOO não esteja concluído, alguns resultados já podem ser apresentados: O Etiquetador Morfológico foi implementado e possui uma taxa de acerto de aproximadamente 95%. Este etiquetador foi construído a partir do zero, visto que não havia nenhum para a língua portuguesa em código aberto. Na literatura encontram-se vários trabalhos relacionados a etiquetadores [1], [2] e [4]; tais sistemas utilizam um *corpus* manualmente etiquetado para extrair as principais regras e generalizá-las para outros textos não etiquetados, conseguindo altas taxas de acerto. Para o treinamento deste módulo, foi utilizado o *Corpus* CETENFOLHA, disponível em [3].

A primeira versão do aplicador de regras já funciona apenas para regras locais, ou seja, aquelas que se aplicam somente a uma vizinhança de 2 ou 3 itens lexicais, sem usar a estrutura da frase. Tanto o detector de sintagmas quanto o detector de relações gramaticais estão funcionando parcialmente.

Até o presente momento foram implementadas regras que cobrem parcialmente os seguintes fenômenos lingüísticos:

- crase (ignorando a análise da regência dos verbos);

- casos de concordância nominal (adjetivo com substantivo na ordem direta, tanto na função de adjunto adnominal como de predicativo de sujeito);
- emprego da expressão invariável "em anexo";
- emprego de "meio" na função de advérbio;
- concordância em gênero entre adjetivo e substantivo no adjunto adnominal;
- flexão do verbo "fazer" indicando tempo;
- flexão do verbo "haver" indicando tempo e indicando existência;
- concordância de artigo com substantivo;
- uso dos pronomes "mim" e "ti";
- uso de "mal" e "mau";
- regência do verbo "preferir".

O corretor gramatical está sendo implementado na linguagem Perl e a interface com o OpenOffice, em Java e C++. Um protótipo já em funcionamento, pode ser acessado no portal do projeto em http://cogroo.incubadora.fapesp.br.

4. Conclusões

O projeto CoGrOO usa uma abordagem híbrida: estatística (por exemplo, o etiquetador morfológico é totalmente estatístico) e baseada em regras. Um dos diferenciais do trabalho apresentado é a abordagem usada na detecção de relações gramaticais, ou seja, a busca por padrões sintáticos no *corpus* anotado. Não foi encontrada abordagem similar na literatura.

A proposta e a estrutura de regras de erros locais usadas neste projeto são baseadas no proposto por [5], contudo este trabalho não trata de questões relacionadas à análise gramatical da sentença, o que o impede de detectar de maneira plena alguns tipos de erros, como os de concordância e regência verbais.

O projeto mais importante de corretor gramatical da língua portuguesa é o ReGra [6], usado no Microsoft Office. Como o projeto CoGrOO ainda está em andamento, não é possível compará-los quanto à taxa de acertos, porcentagem de falsos positivos e falsos negativos.

O cronograma do projeto prevê a disponibilização do código para dezembro de 2005, conforme acordado com o órgão de fomento, mas o protótipo já pode ser testado através do portal criado (http://cogroo.incubadora.fapesp.br).

Agradecimentos

Devem-se profundos agradecimentos a todos os membros da equipe CoGrOO, que muito têm trabalhado para a viabilização do mesmo, entre os quais, Edgard Lemos Jr., Fábio Wang Gusukuma, Marcelo Suzumura, Marcos Yoshio Okamura Oku, William Daniel Colen M. Silva e a professora de Português Sueli Caramello Uliano.

Agradece-se também à FAPESP, Fundação de Amparo à Pesquisa do Estado de São Paulo, pela hospedagem de nosso projeto em sua incubadora (http://incubadora.fapesp.br), e à FINEP, Financiadora de Estudos e Projetos, pelo patrocínio via Edital de Software Livre.

Referências

[1] Brill, E. (1992) "A Simple Rule-Based Part Of Speech Tagger", Proceedings of ANLP-92, 3rd Conference of Applied Natural Language Processing, Trento, Italy.

[2] Daelemans, W., Zavrel, J., Berck, P. and Gillis, S. (1996) "MBT: A Memory-Based Part of Speech Tagger-Generator", In: E. Ejerhed and I. Dagan (eds.) Proceedings of the Fourth Workshop on Very Large Corpora, Copenhagen, Denmark, 14-27.

[3] Linguateca (2005) "CETENFolha", http://www.linguateca.pt/CETENFolha/, acessado em 15/03/2005.

[4] Menezes, C. (2000) "Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos", Dissertação de Mestrado, Universidade de São Paulo, Brazil.

[5] Naber, D. (2003) "A Rule-Based Style and Grammar Checker", Diplomarbeit Technis Fakultät, Universität Bielefeld, Germany.

[6] Nunes, M.G.V.; Oliveira Jr., O.N. (2000) "O processo de desenvolvimento do Revisor Gramatical ReGra", Anais do XXVII SEMISH (XX Congresso Nacional da Sociedade Brasileira de Computação), Volume 1, p.6 (resumo). Artigo Completo na Versão em CD-Rom. PUC-PR, Curitiba, Brazil.

[7] Ueda, R. (2002) "Dicionário br.ispell", http://www.ime.usp.br/~ueda/br.ispell/, acessado em 29/03/2005.

Identificação de *Spam* através de uma Rede Neural *Backpropagation*

Ana C. Bittencourt
Universidade Luterana do Brasil, Ciência da Computação,
Gravataí, Brasil, 94170-240
ana.bittencourt@pop.com.br
e
Sílvia M. W. Moraes[1]
Universidade Luterana do Brasil, Ciência da Computação,
Gravataí, Brasil, 94170-240
silviawm@ulbra.tche.br

Abstract

Today, one of the largest problems for e-mail users is unsolicited messages, also known as *spam*. This work describes the main characteristics of these messages, as well as some used filtering techniques in anti-*spam* tools. Moreover, this paper analyzes and compares the performance of the backpropagation neural network with others used techniques for the identification of spam.

Keywords: *Spam*, Artificial Neural Networks, Automated Text Categorization.

Resumo

Um dos maiores problemas que os usuários de correio eletrônico enfrentam atualmente são as mensagens indesejadas, também conhecidas como *spam*. Este artigo descreve sucintamente as principais características dessas mensagens, bem como algumas técnicas de filtragem normalmente usadas em ferramentas anti-*spam*. E, ainda, analisa e compara o desempenho de uma rede neural *backpropagation* em relação a outras técnicas usadas para a identificação de *spam*.

Palavras-chaves: *Spam*; Redes Neurais Artificiais; Categorização Automática de Texto.

1. INTRODUÇÃO

O termo *spam* é usado para denominar mensagens eletrônicas, em geral, de intuito publicitário, visando promover serviços, produtos ou eventos, que são enviadas sem o consentimento prévio dos destinatários. São mensagens não solicitadas e endereçadas a um grande número de pessoas [1]. O termo UCE (*Unsolicited Commercial Email*) também é usado para indicar mensagens indesejadas, mas quando o conteúdo é exclusivamente de cunho comercial.

Testes realizados revelam que os conteúdos das mensagens classificadas como *spam* são em percentuais: 35% referentes a oportunidades de fazer dinheiro fácil; 11% sobre entretenimento para adultos ou produtos e serviços pornográficos; 10% são marketing direto; 9% contêm guias informativos; 7% oferecem serviços na Internet, promoção de hardware e software e outros produtos para escritório; e 25% referem-se a outros produtos e serviços [2].

Os *spammers*, como são conhecidos os 'entregadores' de *spam*, estão totalmente alheios a qualquer estatística e buscam, cada vez mais, maneiras de invadir a caixa postal de quem quer que seja, ignorando simplesmente o tormento que causam.

[1] A autora Sílvia Maria Wanderley Moraes também é professora da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS).

Há muitos inconvenientes associados ao *spam*. Segundo uma pesquisa realizada sobre o tema, pela empresa de software de segurança Trend Micro, 66% dos entrevistados estão preocupados com a perda de produtividade, bem como com a ameaça de vírus e códigos maliciosos que o *spam* pode representar.

Para as empresas, além da perda de produtividade, existe também o gasto adicional com reforço na segurança das redes, na aquisição de ferramentas anti-*spam* e na contratação de serviços terceirizados para gerenciamento de mensagens. De acordo com o levantamento feito pela empresa de análises de mercado Radicati Group, uma empresa com 10 mil funcionários sem proteção anti-*spam* gasta, em média, US$49 anuais com cada conta de correio eletrônico. A empresa estima também que o processamento dessas mensagens, em 2007, aumentará ainda mais o prejuízo, que crescerá para não menos de US$ 198 bilhões [3].

Para os usuários, os prejuízos vão além da perda de tempo separando mensagens irrelevantes. Eles podem perder mensagens eletrônicas importantes pelo fato de suas caixas postais estarem lotadas, bem como terem seus dados pessoais capturados e seus computadores contaminados por vírus destrutivos. Os *spammers* têm se aproveitado da ingenuidade de grande parte dos usuários, distribuindo cavalos-de-tróia *backdoor* anexados aos *spam*s. Tais arquivos, muitas vezes com aspecto inofensível, utilizam falhas de segurança para invadir sistemas e têm o potencial de criar uma rede de distribuição de mensagens em larga escala sem deixar rastros [4]. Como exemplo de uma falsa mensagem, pode-se citar a que começou a circular pela rede em 22 de abril de 2004, com o nome de '*Bin Laden Captured*', que dizia trazer um vídeo da captura do terrorista, enquanto que na verdade, trazia um cavalo-de-tróia para o computador do internauta [5].

Por estas e muitas outras razões, é importante que as ferramentas de correio eletrônico disponham de filtros que impeçam o *spam*. Algumas ferramentas anti-*spam* identificam mensagens indesejadas pelo remetente e pelo assunto. No entanto, isso é facilmente contornado pelos *spammers* que criam novos remetentes, bem como novos títulos para suas mensagens. Nesse caso, analisar o conteúdo das mensagens, ou seja, seu texto pode ser mais eficiente.

Muitas ferramentas anti-*spam* no mercado já usam tal abordagem. A ferramenta POPFile, da empresa Extravalent, por exemplo, usa o método probabilístico de Bayes. Já a ferramenta AntiVirus for SMTP Gateways 3.1, da empresa Symantec, usa redes neurais na identificação de *spam*. Os bons resultados obtidos com tais ferramentas são motivadores, justificando, inclusive, a pesquisa por métodos eficientes para a análise dos textos de mensagens eletrônicas, bem como de outros documentos.

A exemplo dessas ferramentas, o propósito desse trabalho é analisar o conteúdo das mensagens e classificá-las como *spam* ou legítima[2], através de uma rede neural *backpropagation*. O experimento utiliza técnicas de categorização automática de documentos [6][7] e foi aplicado ao *corpus Ling-Spam*[3]. O artigo, além de analisar os resultados obtidos, compara o seu desempenho com o encontrado por Androutsopoulos que utilizou o método bayesiano em [8].

Este documento está organizado em 4 seções. A seção 2 faz uma breve introdução aos filtros anti-*spam*. A seção 3 descreve trabalhos relacionados que utilizaram outras técnicas para identificar *spam* e apresenta resumidamente o trabalho de Androutsopoulos juntamente com as métricas de avaliação de desempenho usadas pelo autor. E a seção 4 detalha os experimentos realizados com a rede neural, bem como faz um comparativo de desempenho com os experimentos feitos por Androutsopoulos.

2. FILTROS ANTI-SPAM

Há consenso entre os especialistas no que se refere ao problema do *spam*. Eles acreditam que esse problema não pode ser resolvido por ações legais contra os *spam*mers. E indicam como caminho mais viável o uso de filtros anti-*spam*, seja no servidor de correio eletrônico ou no computador do usuário [9]. Tais programas funcionam basicamente de quatro formas: comparando as mensagens que chegam com listas negras *online* ou criadas pelo usuário; liberando apenas as mensagens de remetentes autorizados em sua lista branca; classificando o *spam* por estatística de palavras-chave, pelo método bayesiano; ou mesclando alguns desses mecanismos ou todos [4]. A seguir, uma breve descrição sobre cada um dos métodos de filtragem citados.

- Listas Negras – São bancos de dados de *spam* agrupados, geralmente, pelo endereço IP do *spam*mer [4]. Estão disponíveis na Internet, em sites de entidades sérias que trabalham no combate ao *spam*, como a SpamCop[4], por exemplo. É comum também a definição de listas negras pessoais.

---

[2]Legítima é o termo usado para denominar as mensagens consideradas relevantes para o usuário.

[3] O corpus *Ling-Spam* contém 2.893 mensagens em inglês, sendo 2.412 legítimas e 481 spam. As legítimas são compostas de mensagens lingüísticas enquanto que as spam são mensagens indesejadas recebidas pelo autor (Íon Androutsopoulos). O corpus está disponível em http://www.aueb.gr/users/ion/publications.html

[4] http://www.spamcop.net

- Listas Brancas – Contêm uma relação de endereços cujas mensagens são sempre aceitas [10]. Em ferramentas anti-*spam* que utilizam esse filtro, toda mensagem enviada pode ter seu destinatário automaticamente incluído nessa lista branca, facilitando sua manutenção. A exemplo das listas negras, também podem ser definidas listas brancas pessoais. Essa prática é bastante usual.
- Método Bayesiano – São classificadores estatísticos. Indicam a probabilidade de um determinado termo pertencer a uma classe em particular [11]. Diferentemente dos filtros feitos através de listas, essa técnica analisa o conteúdo da mensagem. Para que o filtro bayesiano classifique as mensagens corretamente é necessário, inicialmente, treiná-lo para reconhecer *spam*. No processo de aprendizagem, é importante que o usuário identifique as mensagens indesejadas. À medida que o treinamento é realizado, a ferramenta forma uma base estatística. Essa base é usada para classificar mensagens futuras.

Existem vantagens e desvantagens em cada um dos métodos. A lista negra, por exemplo, pode deixar passar 'lixos' novos. Na lista branca, como o acesso é restrito apenas aos cadastrados, pode ser que o usuário não receba alguma informação de alguém que esqueceu de cadastrar. E com relação aos filtros estatísticos, apesar de serem menos injustos, também estão sujeitos a falhas.

3. TRABALHOS RELACIONADOS

O trabalho de Androutsopoulos [8] avalia o método *Naive Bayesian* para a filtragem de *spam*. O autor usou em seus experimentos a coleção *Ling-Spam* e investigou, entre outros, a influência do tamanho do conjunto de atributos, do uso da técnica de lematização[5] [12] e de *stop-lists*[6] no desempenho do filtro. O trabalho também apresenta medidas de avaliação sensíveis ao custo bastante interessantes, as quais são descritas logo a seguir na seção 3.1 deste artigo. Androutsopoulos analisou a influência do tamanho do conjunto de atributos realizando testes com conjuntos de 50, 100, 150, 200, ... , 650 até 700 atributos. Os atributos dos conjuntos foram selecionados através da técnica de ganho de informação [13]. O experimento apresentou bons resultados com conjuntos de 50,100, 200 e 300 atributos.
Já, para verificar a influência do uso de lematização e de *stop-lists*, o autor configurou o filtro de quatro formas diferentes no que tange ao preparado das mensagens, as quais chamou de:

- *bare*: sem remoção de *stopwords* [7] e sem aplicação da técnica de lematização;
- *lemm*: aplicação da técnica de lematização sem remoção de *stopwords*;
- *stop*: remoção de *stopwords* e sem aplicação da técnica de lematização;
- *lemm_stop*: remoção de *stopwords* e aplicação da técnica de lematização;

Os resultados dos experimentos descritos pelo autor indicaram que a técnica de lematização melhorou o desempenho do filtro. No entanto, o uso da *stop-list* não colaborou de forma significativa para evitar classificações incorretas de mensagens legítimas como *spam.* Apesar disso, o método *bayesiano* apresentou um alto índice de precisão e abrangência. Quando o custo de bloquear uma mensagem legítima é baixo, o filtro tem uma contribuição positiva e significante.
Mais tarde, em um novo trabalho [14], Androutsopoulos comparou o método bayesiano com a técnica de abordagem baseada em memória. Nesse trabalho, Sakkis e Androutsopoulos usaram extensões do algoritmo do vizinho mais próximo (*k-Nearest-Neighbor*) e avaliaram o desempenho do filtro em relação ao comprimento da vizinhança, do conjunto de atributos e do *corpus* de treinamento. Os resultados encontrados pelo autor foram mais satisfatórios do que com o método bayesiano. A técnica de abordagem em memória apresentou melhor desempenho na maioria dos casos.
Gee apresenta também em [15] o método *Naive Bayesian* para filtragem de *spam*, porém aplicando a técnica de *semântica latente* [6] na fase de seleção de características. Dentre as conclusões do autor, pode-se destacar que semântica latente é um método viável para classificar *spam*, por melhorar o percentual de abrangência das mensagens analisadas e classificadas como *spam;* e que o uso da técnica de lematização é recomendável.
Pode-se citar, ainda, outros trabalhos como de Drucker, Wu e Vapnik [16] que estudaram o método *máquinas de vetor de suporte* (SVM) para classificação de *spam*. Os autores compararam SVM com os algoritmos *Ripper*, *Rocchio* e árvores de decisão *boosting*. Segundo os autores as árvores de decisão e a SVM tiveram bom desempenho no que se

---

[5]A técnica de lematização reduz as variantes morfológicas de uma palavra para a sua forma base. O infinitivo é usado para representar as formas do paradigma verbal, enquanto o masculino singular representa o paradigma nominal e adjetivos.

[6]*Stop-list* é uma lista de termos (normalmente palavras) considerados irrelevantes, principalmente, por serem muito freqüentes nos textos. Em geral, as *stop-list* contêm artigos, pronomes, preposições, etc.

[7] *Stopwords* são termos de uma *stop-list*. *A stop-list* usada contém os 100 termos mais frequentes do *British National Corpus* (BNC), e está disponível em ftp://ftp.itri.bton.ac.uk/pub/bnc.

refere à precisão e velocidade. No entanto, a SVM apresentou menor tempo de treinamento. Outra constatação dos autores é que para identificação de *spam* é recomendável não usar *stop-list*.

3.1 Métricas de Avaliação de Desempenho

Nesta seção é apresentada a forma de avaliação de desempenho usada Androutsopoulos [8]. O experimento de identificação de *spam* por uma rede neural descrito neste trabalho também utilizou tais medidas a fim de viabilizar a comparação dos resultados.

Androutsopoulos avalia os resultados do seu trabalho através de medidas de precisão (SP), abrangência (SR), acurácia (Acc) e taxa de erro (Err). A precisão define o percentual de exatidão de classificação das mensagens. A abrangência mede a relação entre o número de mensagens corretamente associadas a uma classe e o número de mensagens que realmente pertencem à essa classe. A acurácia define a relação entre a quantidade total de mensagens corretamente classificadas e o total de mensagens utilizadas. A taxa de erro mede a quantidade total de mensagens erroneamente classificadas em relação ao total de mensagens da base. As fórmulas a seguir descrevem o cálculo das medidas citadas:

$$SP = \frac{n_{S\to S}}{n_{S\to S} + n_{L\to S}},\ SR = \frac{n_{S\to S}}{n_{S\to S} + n_{S\to L}},\ Acc = \frac{n_{L\to L} + n_{S\to S}}{N_L + N_S} \text{ e } Err = \frac{n_{L\to S} + n_{S\to L}}{N_L + N_S}$$

Onde, $n_{L\to S}$ refere-se à quantidade de mensagens legítimas classificadas como *spam*; $n_{S\to L}$, à quantidade de *spam*s classificadas como legítimas; $n_{S\to S}$, à quantidade de *spam*s classificadas corretamente; $n_{L\to L}$, à quantidade de legítimas classificadas corretamente; N_L , à quantidade total de legítimas e N_S , à quantidade total de *spam*s.

Em [8] são usadas também medidas referentes ao custo de se classificar incorretamente as mensagens. Classificar uma mensagem legítima como *spam* (L→S) ou classificar uma mensagem *spam* como legítima (S→L) são dois tipos de erros. No entanto, bloquear uma mensagem legítima (classificada como *spam*) é muito mais caro do que permitir que uma mensagem *spam* passe pelo filtro (classificada como legítima). Por essa razão, o autor atribuiu um limiar (λ) ao classificador, que determina que classificar incorretamente L→S é λ vezes mais caro que S→L.

Para tornar as medidas de acurácia e taxa de erro sensíveis ao custo, o autor trata cada mensagem legítima como λ mensagens. Desta forma, quando uma mensagem legítima é mal classificada, conta-se como λ erros; e quando é classificada corretamente, conta-se como λ sucessos. A partir disso, o peso da acurácia e peso da taxa de erro ($WErr = 1 - WAcc$) podem ser definidos como:

$$WAcc = \frac{\lambda \times n_{L\to L} + n_{S\to S}}{\lambda \times N_L + N_S} \qquad WErr = \frac{\lambda \times n_{L\to S} + n_{S\to L}}{\lambda \times N_L + N_S}$$

Quando se usa acurácia ou taxa de erro (com peso ou não), é importante comparar com o método "baseline". De acordo com o autor, "baseline" representa a completa ausência de filtro, ou seja, nenhuma mensagem legítima é bloqueada e todas as mensagens *spam* simplesmente passam pelo filtro. Assim, pode-se determinar se a presença do filtro é de fato necessária e, portanto, se o custo computacional de processamento das mensagens é justificado. O peso da acurácia e taxa de erro "baseline" é calculado por:

$$WAcc^b = \frac{\lambda \times N_L}{\lambda \times N_L + N_S} \qquad WErr^b = \frac{N_S}{\lambda \times N_L + N_S}$$

Para comparar com o "baseline", calcula-se a taxa de custo total (TCR):

$$TCR = \frac{WErr^b}{WErr} = \frac{N_S}{\lambda \times n_{L\to S} + n_{S\to L}}$$

Altos índices de TCR indicam alta performance. Para TCR < 1(menor que um), é melhor não utilizar o filtro. O custo é proporcional ao tempo gasto. A medida TCR avalia quanto tempo é gasto para apagar manualmente todos *spam*s quando não houver filtro algum (N_s), comparado com o tempo gasto para apagar manualmente algum *spam* que passe pelo filtro ($n_{S\to L}$) mais o tempo necessário para recuperar mensagens legítimas bloqueadas erroneamente ($\lambda \times n_{L\to S}$) [8].

4. O EXPERIMENTO

Para realização deste experimento foi desenvolvido um protótipo que implementa como filtro anti-spam uma rede neural *Perceptron* Multicamadas, com o algoritmo de aprendizagem *backpropagation* [17]. Cabe ressaltar que o protótipo ainda não foi integrado a uma ferramenta de correio eletrônico.

A identificação das mensagens é feita pela análise do seu conteúdo, utilizando a categorização de textos para realizar essa tarefa, mais especificamente, utilizando redes neurais como método de categorização. O motivo dessa escolha foi, principalmente, por não terem sido encontradas no mercado muitas ferramentas anti-*spam* que utilizassem redes neurais artificiais (RNA). A maioria usa métodos estatísticos para classificar *spam*.

Para a seleção de características, a técnica escolhida foi a de escore de relevância, por apresentar bons resultados em [18] e [7]. O experimento foi aplicado à coleção de mensagens Ling-*Spam* com o propósito de viabilizar a comparação dos resultados com trabalhos relacionados, tais como [8]. Do total de mensagens da coleção, 70% foi utilizado para a fase de treinamento e os 30% restantes para a fase de testes da RNA.

Como já mencionado, a coleção Ling-*Spam* foi organizada em quatro subcoleções (configurações do filtro). Em seus experimentos, Androutsopoulos não removeu os caracteres especiais, mas desprezou anexos e *tags html* e, ainda, acrescentou o assunto de cada mensagem no seu texto.

4.1 Análise das Mensagens

O protótipo implementado para realizar o experimento analisa as mensagens do *corpus* em três etapas: pré-processamento, RNA e pós-processamento. A fase de pré-processamento realiza o tratamento e a formatação das mensagens para que essas possam ser analisadas pela RNA. Na fase seguinte, a RNA já treinada processa as mensagens e gera duas saídas. Essas saídas são analisadas pela fase de pós-processamento, através da qual as mensagens são, então, classificadas como *spam* ou legítima; ou não são classificadas.

4.1.1 Pré-Processamento

Como no *corpus* utilizado as mensagens já estavam semi-preparadas (em algumas bases, as *stopwords* já tinham sido removidas e a técnica de lematização também já tinha sido aplicada) foi necessário apenas acrescentar mais algumas opções. Acrescentou-se, com o objetivo de analisar os resultados, a opção de remoção de caracteres especiais e o uso do cálculo de similaridade durante o processo de extração dos termos.

O algoritmo utilizado para calcular a similaridade foi o *soundex*, que faz uma análise fonética da palavra, buscando semelhanças quanto à pronúncia delas.

Para cada subcoleção testada, foi necessário realizar o processo de identificação dos termos da base. O protótipo considerou como termo qualquer seqüência de caracteres que estivesse entre espaços em branco. Cabe ressaltar que optou-se por excluir os números, simplesmente porque o protótipo não tratava termos compostos, e também porque números para terem significado precisam estar contextualizados. Além disso, os termos foram convertidos para caixa alta, a fim de impedir que termos iguais fossem identificados como diferentes.

Para que as mensagens, que são constituídas de termos simbólicos, pudessem ser processadas pela RNA, que é numérica, foi necessário construir um vetor base. Este vetor determina o formato dos padrões de entrada da rede neural. O vetor base é formado pelas *N* características mais significativas da base, conforme mostra a Figura 1.

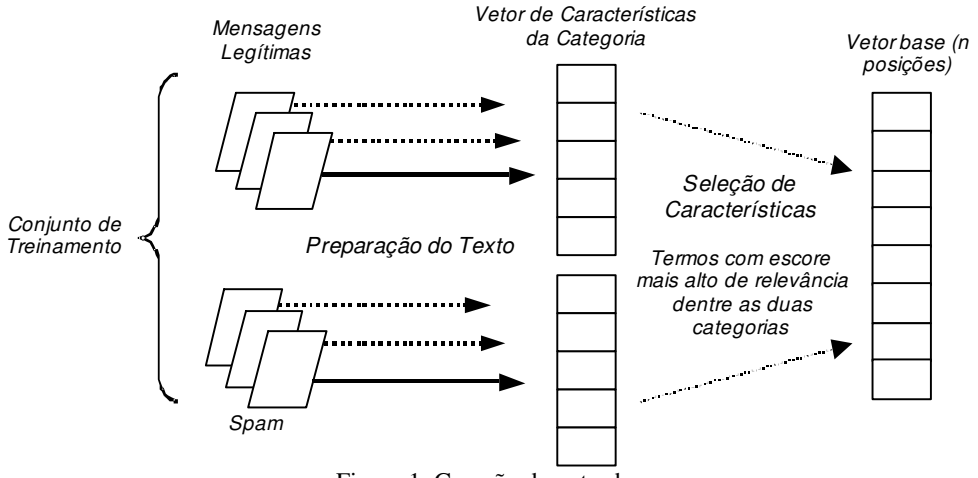


Figura 1. Geração do vetor base

Com o objetivo de comparar com os melhores resultados obtidos em [8], foram construídos vetores base com 50, 100, 200 e 300 características para cada subcoleção testada.
Após a definição do vetor base, as mensagens foram formatadas. Cada mensagem legítima ou não, após a fase de identificação de termos e seleção de características, passou pela etapa de definição do seu vetor de treinamento. O vetor de treinamento é um vetor binário com a mesma dimensão do vetor base. Ele é gerado, comparando o vetor de características da mensagem com o vetor base (vide Figura 2).
O vetor de treinamento terá o valor 1 na posição *i*, se a característica *i* do vetor base estiver presente dentre as características da mensagem. Ele receberá 0 (zero) nas posições correspondentes às características do vetor base ausentes na mensagem. No exemplo da Figura 2, os termos SEX, DOLAR e OFFER do vetor de características, também aparecem no vetor base, logo, recebem o valor 1 no índice *i* correspondente do vetor de treinamento, enquanto que os termos HOUSE, PAPER e BOOK recebem o valor 0 (zero), por não serem encontrados no vetor base. Após a definição dos vetores de treinamento para cada mensagem da subcoleção em análise, a rede pode iniciar o seu processamento, tanto para aprender o padrão dessas mensagens quanto para identificá-los.

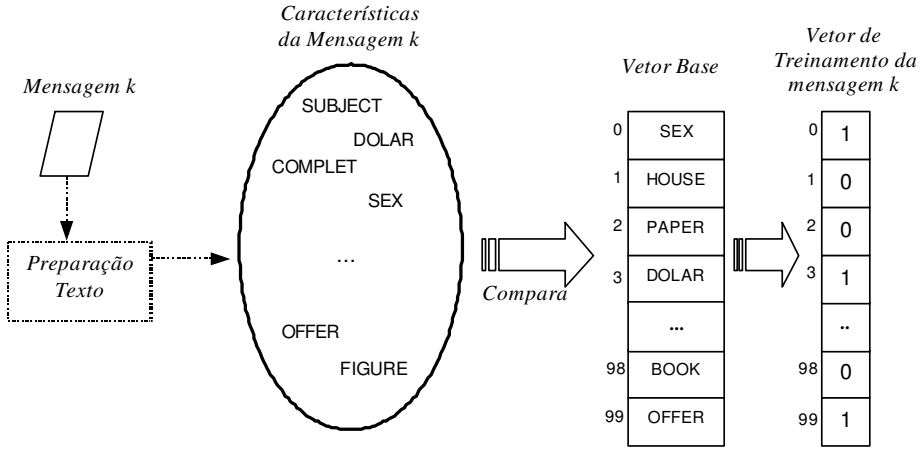


Figura 2. Geração do vetor de treinamento

4.1.2 RNA
Como já mencionado, o tipo de RNA utilizado foi o *Perceptron* Multicamadas, com o algoritmo de aprendizagem *backpropagation*. Através do *MATLAB*[8], diversos testes foram feitos para definir qual o número de neurônios a ser utilizado na camada intermediária, já que o número de neurônios da camada de entrada já estava definido em *N* (número de atributos determinados pelo vetor base) e a camada de saída em 2 (dois) neurônios: um representando a classe *spam* e o outro, a classe mensagens legítimas (conforme modelo apresentado na Figura 3). Os testes foram realizados com 4, 8, 10, 14, 16, 20 e 24 neurônios na camada intermediária e observou-se que 8 neurônios foram suficientes para a obtenção de resultados satisfatórios na classificação, como pode ser visto em [19].

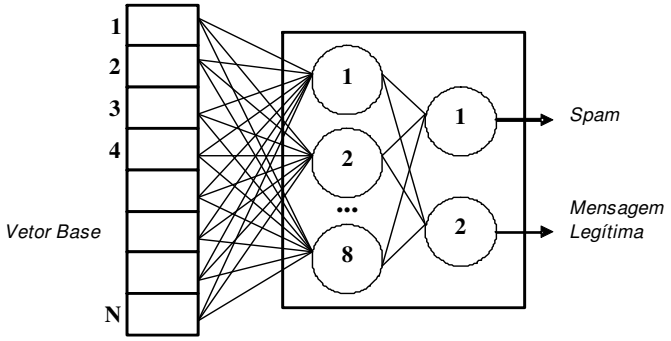


Figura 3. Modelo da topologia da RNA

[8] Software interativo de alta performance voltado para o cálculo numérico. Integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos.

Estando os vetores de treinamento devidamente construídos, o processo de aprendizagem foi realizado com 70% das mensagens de cada subcoleção em análise, ficando as demais para serem usadas durante o teste da rede. A rede implementou como função de ativação a tangente hiperbólica em todas as camadas devido aos bons resultados obtidos.
A rede foi treinada a uma taxa de aprendizagem de 0.1 e foram usados como critérios de convergência o erro médio quadrado (EMQ) e o número de épocas. Depois de alguns de testes, foi determinado que quando o EMQ tivesse uma variação inferior a 10^{-4} de uma época para outra ou já tivessem transcorrido 100 épocas, a rede teria seu processo de aprendizagem encerrado. A Tabela 1 abaixo descreve o número médio de épocas em que a rede convergiu, conforme o número de termos, para a coleção *stop*.

Tabela 1. Treinamento coleção *stop*

Termos	Épocas
50	79
100	75
200	70
300	63

4.1.3 Pós-Processamento
A fase de Pós-Processamento só é executada durante a etapa de generalização da rede. É nessa fase que se analisa a aprendizagem realizada pela rede. Para esta fase foram usadas 30% das mensagens de cada base. Essas mensagens não foram utilizadas para treinar a rede. Os vetores de teste são gerados da mesma forma que os de treinamento.
Cabe ressaltar que o vetor base usado nessa fase é o mesmo da fase de treinamento para cada uma das subcoleções. Esses vetores, ao passarem pela rede já treinada, ativam os neurônios de saída da rede. O pós-processamento é justamente a análise das saídas geradas pela rede a fim de determinar se a mensagem é *spam*, legítima ou se a RNA não conseguiu classificá-la.
A rede classifica a mensagem analisando os resultados gerados pelos seus neurônios da camada de saída. O neurônio que tiver a maior saída contínua é escolhido como a categoria da mensagem. No entanto, apenas esse critério não é suficiente, pois como as saídas dos neurônios são contínuas, a rede pode gerar saídas muito parecidas. Ela pode, por exemplo, gerar no neurônio referente a *spam* um resultado de 0.79 e no neurônio de mensagem legítima 0.81. Usando apenas o critério inicial, a mensagem seria classificada como legítima, mas analisando-se as saídas percebe-se que isso pode gerar um erro de classificação visto que o resultado do neurônio *spam* também foi significativo.
Para resolver este problema foi definido um segundo critério. Decidiu-se que quando a diferença entre as saídas dos dois neurônios fosse igual ou inferior a 0.2 a rede não classificaria a mensagem. Chegou-se a este valor após alguns testes e análises dos resultados, viu-se que grande parte das mensagens que classificavam erroneamente, tinha aproximadamente essa diferença entre as duas saídas. Em função disso, optou-se por não classificar a mensagem do que classificá-la incorretamente.

4.2 Avaliação de Desempenho da Rede *Backpropagation*
Os resultados obtidos no experimento com a rede *backpropagation* são mostrados a seguir na Tabela 2. Comparando os resultados obtidos com os de [8], observou-se que o percentual de abrangência da rede neural foi maior em todos os casos. No entanto, o método bayesiano foi mais preciso. Observou-se também que para os três valores de λ: 1, 99 e 999, definidos por Androutsopoulos em [8], o TCR mais alto obtido pela rede foi com a *stoplist* habilitada, tendo um ganho significativo com relação ao método utilizado por [8].
Dentre as constatações observadas quanto ao desempenho da rede *backpropagation*, pode-se destacar que a remoção das *stopwords* do texto melhorou o desempenho do filtro. Nos testes realizados (vide Tabela 3), a eliminação das *stopwords* aumentou o índice de classificação correta para 99,59% no caso de mensagens legítimas e para 93,79% em mensagens *spam* (coleção *stop* utilizando 200 termos). Os resultados obtidos indicam, ainda, que a eliminação desses termos também reduz o número de falsos positivos e falsos negativos.
Além disso, com essa configuração, a rede conseguiu índices *TCRs* mais altos, mesmo para limiares maiores (λ=*999*). Um resultado bastante animador é que a rede com a configuração *stop* conseguiu superar os resultados encontrados em[14], que utiliza o algoritmo do vizinho mais próximo, cujos melhores resultados são apresentados na Tabela 4.
Observou-se, ainda, que o uso de um número maior de características nos vetores de entrada da rede neural resultou em um melhor desempenho de classificação, e isso foi constatado ao se aumentar o número de características para 200 e 300 termos. As coleções *lemm* e *lemm_stop* foram as que, em uma média geral, melhor classificaram *spam*, porém tiveram índices mais baixos na classificação de mensagens legítimas. Para essas configurações o método bayesiano obteve melhores resultados, conforme apresentado na tabela 5.

Tabela 2. Resultados obtidos pela rede *backpropagation*

Configuração de Filtro	λ	Termos	SR	SP	Wacc	$WAcc^b$	TCR
(a) bare	1	50	99,27%	93,79%	95,857%	83,314%	14,50
(b) stop-list	1	50	99,28%	94,52%	96,778%	83,314%	16,11
(c) lemmatizer	1	100	99,29%	92,67%	88,723%	83,314%	12,08
(d) lemmatizer + stoplist	1	100	100,0%	92,81%	89,873%	83,314%	13,18
(a) bare	9	200	94,41%	98,54%	99,580%	97,823%	05,58
(b) stop-list	9	200	97,14%	100,0%	99,460%	97,823%	36,25
(c) lemmatizer	9	100	99,29%	92,67%	87,479%	97,823%	01,45
(d) lemmatizer + stoplist	9	100	100,0%	92,81%	88,470%	97,823%	01,47
(a) bare	999	200	94,41%	98,54%	99,722%	99,979%	00,07
(b) stop-list	999	200	97,14%	100,0%	99,584%	99,979%	36,25
(c) lemmatizer	999	300	95,00%	99,25%	99,860%	99,979%	00,14
(d) lemmatizer + stoplist	999	300	92,20%	99,24%	99,584%	99,979%	00,14

Tabela 3– Comparação dos resultados da configuração *stop* para Bayes[8] *e RNA*

Técnica	λ	Termos	SR	SP	TCR
Método Bayesiano	1	50	82,35%	97,13%	14,50
RNA	1	50	99,28%	94,52%	16,11
Método Bayesiano	9	200	76,11%	99,47%	12,08
RNA	9	200	97,14%	100,00%	13,18
Método Bayesiano	999	200	73,40%	99,43%	05,58
RNA	999	200	97,14%	100,0%	36,25

Tabela 4 – Resultados do algoritmo k-NN [14]

K	λ	Termos	SR	SP	TCR
8	1	600	88,60%	97,37%	7,18
2	9	700	81,93%	98,79%	3,64
7	999	600	59,91%	100,00%	2,64

Tabela 5 – Comparação dos resultados da configuração *lemm* para Bayes[8] *e RNA*

Técnica	λ	Termos	SR	SP	TCR
Método Bayesiano	1	100	82,35%	99,02%	5,41
RNA	1	100	99,29%	92,67%	12,08
Método Bayesiano	9	100	77,57%	99,45%	3,82
RNA	9	100	99,29%	92,67%	1,45
Método Bayesiano	999	300	63,67%	100,00%	2,86
RNA	999	300	95,00%	99,25%	0,14

Tendo em vista os bons resultados obtidos com a rede neural, foram realizados mais testes para identificar se a remoção de caracteres especiais do texto da mensagem, bem como se o uso de uma função de similaridade na construção dos vetores de treinamento aumentariam o desempenho do filtro. Tais modificações resultaram em bons índices para classificação de mensagens legítimas, chegando a ter 100% de acerto na classificação (coleção *lemm_stop* com 300 termos), no entanto, obtiveram um percentual de acerto para a classificação de *spam* de apenas 86,90%. Este resultado, se comparado com outros, não é muito motivador, pois além de filtrar menos as mensagens indesejadas ainda tem a desvantagem de agregar mais tempo de processamento ao protótipo. Mais detalhes sobre os testes realizados podem ser encontrados em [19].

5. CONSIDERAÇÕES FINAIS

De acordo com o estudo realizado, pode-se concluir que não existe, ainda, uma solução efetiva para o *spam*. No entanto, filtrá-lo pode ser uma boa saída para combatê-lo. A pesquisa deixa evidente que filtros que analisam o conteúdo das mensagens obtêm, em geral, melhores resultados. Embora o método bayesiano seja usado com mais freqüência pelas ferramentas anti-*spam*, o método de categorização por rede neural *backpropagation* também apresentou bons resultados, o que motiva a continuidade desta pesquisa.

Apesar dos resultados alcançados serem satisfatórios, acredita-se que com a aplicação de outras técnicas para seleção de características, a rede neural possa alcançar ainda melhores resultados. Desta forma, pretende-se testar outras técnicas de seleção de características, bem como outros modelos de RNAs e, ainda, integrar o protótipo a uma ferramenta de correio eletrônico.

Referências

[1] Jessen, K. S.; Chaves, M. H. P. C. e Hoepers, C. Projeto e Desenvolvimento de um Sistema de Controle e Acompanhamento e Notificações de Spam. *V Simpósio Segurança em Informática (SSI'2003)*, (Novembro, 2003).

[2] Campos, J; Monteiro, E. Um Agente de Filtragem de Correio Electrônico Indesejado. Coimbra, Portugal: 1998.

[3] Folha Online, 10 jun. 2003. Disponível em: <http://www1.folha.uol.com.br/folha/informatica/ult124u13138.shtml>. Acesso em: 16 abr. 2004.

[4] Reggiani, L. O submundo dos spammers. *INFO Exame*, São Paulo, SP, ano 18, nº 211, (Outubro, 2003), pp. 46-52.

[5] Plantão INFO, 23 abr. 2004. Disponível em: <http://info.abril.com.br/aberto/infonews/042004/23042004-0.shl>. Acesso em: 04 mai. 2004.

[6] Sebastiani, F. A Tutorial on Automated Text Categorization. Pisa: CNR, Instituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, 1999.

[7] Galho, T. S. e Moraes, S. M. W. Categorização Automática de Documentos de Texto Utilizando Lógica Difusa. *Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação)*, Centro de Ciências Naturais e Exatas, Universidade Luterana do Brasil. Gravataí: 2003.

[8] Androutsopoulos, Ion et al. An Evaluation of Naive Bayesian Anti-Spam Filtering. *Software and Knowledge Engineering Laboratory, National Centre for Scientific Research 'Demokritos'*. Athens, Greece: 2000.

[9] Grego, M. Fora, chatos! *INFO Exame*, São Paulo, SP, ano 19, nº 221, (Agosto, 2004), pp.39-41.

[10] Costa, E. Xô, porcaria! 7 Ferramentas para expulsar o spam da sua rotina. *INFO Exame*, São Paulo, SP, ano 18, nº 211, (Outubro, 2003), pp. 58-60.

[11] Han, J. e Kamber, M. *Data Mining: Concepts and Techniques*. San Francisco, CA: Morgan Kaufmann, 2001.

[12] Moens, M. Automatic indexing and abstract of documens texts, Massachusetts: Kluwer Academic Publishers, 2000.

[13] Yang, Y.; Pedersen, J. A Comparative Study on Feature Selection in Text Categorization. <http://www.cs.cmu.edu/~yiming/publications.html>, 1997.

[14] Sakkis, Georgios. et al. A Memory-Based Approach to Anti-spam Filtering for Mailing Lists, Kluwer Academic Publishers, Information Retrieval, v. 6, p. 49-73, jan, 2003.

[15] Gee, K. Using Latent Semantic Indexing to Filter Spam, Texas: The University of Texas at Arlington, Department of Computer Science and Engineering, 2003.

[16] Drucker, H..; Wu, D.; Vapnik, V. Support Vector Machines for Spam Categorization, IEEE Transactions on Neural Networks, v. 10, n. 5, p. 1048-1054, set, 1999.

[17] Haykin, S. *Redes Neurais – Princípios e Práticas*. Bookman, 2000.

[18] Rizzi, C. B. Categorização de Textos por Rede Neural Estudo de Caso. *Dissertação (Mestrado em Ciência da Computação)*, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre: 2000.

[19] Bittencourt, A. C Identificação de Spam através de uma Rede Backpropagation. Gravataí: ULBRA. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação), Centro de Ciências Naturais e Exatas, 2004.

Aplicación de Tecnología de Agentes Asistentes de Usuario para la Resolución de Consultas en Repositorios de Datos Genómicos

David Benaderet
Universidad Católica del Uruguay, Facultad de Ingeniería y Tecnologías,
Montevideo, Uruguay, 11600
dbenader@adinet.com.uy

y

Ana Janauskas
Universidad Católica del Uruguay, Facultad de Ingeniería y Tecnologías
Montevideo, Uruguay, 11600
ajanauskas@netgate.com.uy

y

Ernesto Ocampo Edye
Universidad Católica del Uruguay, Facultad de Ingeniería y Tecnologías
Montevideo, Uruguay, 11600
eocampo@ucu.edu.uy

Abstract

This article is framed in the Bioinformatic field, and intends to show up the problems of the users of genomic data repositories, when they try to solve complex queries using these repositories. Also, this paper will exposes a hypothesis to solve the referred problems, which is based in the application of user assistant agent technology to the bioinformatics queries resolution systems. This hypothesis was tested by constructing a multiagent system that allowed users to perform complex bioinformatics queries. Contrasting the advantages and disadvantages of the developed system, with the advantages and disadvantages of the current way in which the queries are solved, we have concluded that the application of intelligent agents for the development of a query resolution bioinformatic system, allows the user abstraction of the bioinformatic tools to solve the query and gives an easy-to-use interface, providing useful assistance during the system utilization.

Keywords: Bioinformatics, Multiagent systems, User assistant agents, User assistance, Genomic data repository, Complex queries resolution.

Resumen

El presente artículo se enmarca en el área de la Bioinformática, y pretende poner de manifiesto los diferentes problemas que atraviesan los usuarios de repositorios de datos genómicos a la hora de realizar consultas complejas sobre los mismos. Asimismo, expone una hipótesis que intenta resolver los problemas referidos, la cual se basa en aplicar tecnologías de agentes asistentes de usuarios a los sistemas de resolución de consultas bioinformáticos. Dicha hipótesis fue sometida a prueba mediante la construcción de un sistema multiagente, que permitió a los usuarios realizar consultas bioinformáticas complejas. Del contraste entre las ventajas y desventajas del sistema desarrollado, con las ventajas y desventajas de la forma actual en que las consultas son resueltas, concluimos que la aplicación de agentes inteligentes para el desarrollo de un sistema bioinformático de resolución de consultas, permite abstraer al usuario de las herramientas bioinformáticas utilizadas para resolver las mismas y brinda una interfaz sencilla de utilizar, proveyendo de asistencia útil durante la utilización del sistema.

Palabras claves: Bioinformática, Sistemas multiagentes, Agentes asistentes de usuario, Asistencia a usuarios, Repositorios de datos genómicos, Resolución de consultas complejas.

1. INTRODUCCIÓN

La realidad del biólogo de hoy ha cambiado con respecto a la del biólogo veinte años atrás. En aquel entonces la secuenciación de pequeñas porciones de proteínas o ADN tomaba meses de trabajo, situación que en conjunción con el carácter privado que se daba a los resultados obtenidos, no hacía pensar en la necesidad de métodos de almacenamiento específicos y mucho menos de contar con un estándar de representación de los datos [1].
En el presente la secuenciación ha dejado de ser un problema. El empleo por parte de los distintos laboratorios de nuevas técnicas analíticas hace que el caudal de información producida día a día sea enorme, por lo que el volumen acumulado ha crecido de forma explosiva. La información producida no es de carácter privado, sino que se comparte a través de sitios públicos en Internet.
Como consecuencia los biólogos u otros potenciales usuarios pueden pensar en estudios y proyectos más ambiciosos, ya que disponen de abundante información. Sin embargo, la propia abundancia de información biológica presenta un problema en sí misma.
El creciente aumento de información genómica que ocurre día a día, el carácter distribuido de dicha información, y la necesidad por parte de usuarios no expertos de acceder a la misma, imponen la necesidad de contar con sistemas informáticos, los cuales hagan posible al usuario acceder de forma sencilla a la información disponible.

2. PLANTEO DEL PROBLEMA

Las necesidades de los usuarios interesados en información biológica, con respecto a la manipulación y consulta de las bases de datos genómicas y proteómicas son muy variadas, existiendo varios sistemas públicos accesibles por la Web para satisfacer dichas necesidades de información y consulta.
El problema actual es que los análisis de la información disponible se vuelven cada vez más complejos. La existencia de múltiples repositorios públicos con diferente información, nivel de procesamiento de sus datos y técnicas para consultarlos, hace necesario realizar consultas usando distintas herramientas sobre varios repositorios.
Por tanto, para ser eficientes, los biólogos deben familiarizarse con una multitud de herramientas y formatos de datos, cuyo conocimiento está por fuera de la órbita del biólogo.
Las siguientes dificultades se le presentan al usuario al utilizar las herramientas públicas existentes a través de Internet, para resolver sus necesidades de información:

- Necesidad de conocer distintas interfaces complejas de dominar
- Necesidad de realizar las mismas consultas sobre distintos repositorios
- Necesidad de utilizar varios sistemas diferentes para la resolución de una única consulta
- Necesidad de obtener conocimiento bioinformático avanzado

3. HIPÓTESIS DEL TRABAJO

La utilización de agentes inteligentes para el desarrollo de un sistema bioinformático de consultas, permite abstraer al usuario de las herramientas bioinformáticas utilizadas para resolver las consultas, y brindarles una interfaz sencilla de utilizar, proveyéndolos de asistencia útil durante la utilización del sistema.

4. OBJETIVOS DE LA INVESTIGACIÓN

El objetivo principal de la investigación consiste en resolver la dificultad que tienen los usuarios con respecto a tener que conocer distintas interfaces complejas de diversos sistemas y poseer conocimiento bioinformático para poder utilizar las herramientas existentes.

5. PROPUESTA

El siguiente trabajo se centra en el estudio y diseño de agentes asistentes de usuario para lograr los objetivos planteados. Los agentes asistentes de usuario son las entidades que atienden al usuario cuando ingresa al sistema y lo guían a lo largo de la utilización del mismo. Estos componentes le brindan asistencia personalizada, seleccionan la manera más óptima de resolver sus consultas, y delegan las tareas de resolución a los componentes apropiados.
El sistema asistente permite abstraer al usuario de las herramientas bioinformáticas utilizadas para resolver las consultas, a través de una interfaz sencilla de utilizar.
A través de esta solución se intenta hacer posible para distintos usuarios ajenos a la bioinformática la utilización de la información biológica existente.

5.1 Requerimiento específico.

Con el objetivo de acotar el dominio del problema, para poder probar la aplicación de la tecnología de agentes en las aplicaciones bioinformáticas, se definió un requerimiento específico el cual es resuelto por el sistema implementado.
Tal requerimiento consiste en encontrar patrones diferenciales en cierta especie con respecto a la susceptibilidad o

resistencia a cierto patógeno.

6. VISIÓN GENERAL DEL SISTEMA

La aplicación desarrollada consiste en un sistema multiagente de consulta e integración de información genómica.
Dentro de la aplicación se pueden distinguir seis roles claves:

- Asistencia a Usuarios: asiste al usuario durante su estancia en el sistema. Permite individualizar a cada usuario y adaptar el sistema a las preferencias de los mismos.
- Planificación de Consultas: establece los pasos para resolver una consulta bioinformática determinada.
- Ejecutor de Consultas: ejecuta las consultas bioinformáticas accediendo al repositorio de datos local, y utilizando las herramientas bioinformáticas disponibles localmente.
- Coordinador de Repositorio Local: posee información relacionada al repositorio local y a los datos que existen en el mismo.
- Acceso a Sistemas Externos: accede a repositorios de datos de ESTs[1] y a herramientas bioinformáticas externas, a través de Internet.
- Adaptador de Base de Datos: accede a repositorios de ESTs disponibles a través de Internet y vuelca su contenido en el repositorio de datos de local.

En un extremo del sistema se encuentran los usuarios. Cuando un usuario accede al sistema es atendido por un Asistentes de Usuario, el cual lo ayuda a utilizar el sistema, adaptándose a sus preferencias.
La Planificación de la Consulta tiene por objetivo desarrollar el plan más apropiado para la resolver la consulta que el usuario realiza. Este rol determina si la información necesaria para cada etapa del plan, es más conveniente obtenerla del repositorio local o de sistemas externos.
Si el Planificador de Consultas considera apropiado obtener los datos del repositorio local, se delega la ejecución al Ejecutor de Consultas, en cambio, si considera obtenerla de repositorios externos, delega la ejecución mediante el Acceso a Sistemas Externos.
El Ejecutor de Consultas consulta datos del repositorio local, para lo cual se comunica con el Coordinador del Repositorio Local, el cual se encarga de brindarle la información y datos que se requieran.
El rol de Acceso a Sistemas Externos tiene la capacidad de consultar repositorios genómicos externos y utilizar herramientas bioinformáticas externas.
Por último, el repositorio local se nutre de forma permanente de datos sobre ESTs que se encuentra en repositorios de datos accesibles mediante Internet. A tales efectos existen los Adaptadores de Bases de Datos, que extraen datos de los diferentes repositorios de información, adaptándolos para poder integrarlos al repositorio local.
En el presente trabajo de investigación fueron estudiados, diseñados, e implementados, los dos primeros módulos (Asistencia a Usuarios y Planificación de Consultas), los cuales se enmarcan en el área de la asistencia al usuario. Por tal motivo, en el presente artículo se estudiarán los mencionados módulos los cuales se incluyen en el denominado Subsistema Asistente.

6.1 Sistema Asistente

Los agentes asistentes son responsables de supervisar las acciones de los usuarios para proporcionar ayuda a los mismos, de tal forma que las consultas ingresadas sean resueltas de forma óptima, liberándolos de ejecutar tareas repetitivas y de poseer conocimientos técnico, ya sea en el ámbito de la informática como en el de la biología.
Estos agentes requieren de un motor de inferencia que les permita a través de la percepción de las evaluaciones del usuario, inferir la forma más óptima de resolver la consulta que se les plantea.
Para cumplir con este requerimiento, los agentes están dotados de la capacidad de aprendizaje, para desarrollar su conocimiento sobre como resolver las consultas de la mejor manera. Este conocimiento es representado a través de la base de conocimiento del sistema asistente.
Para brindar asistencia personalizada a los usuarios, los agentes asistentes se encargan de almacenar datos de los usuarios, constituyéndose un modelo de usuarios, para poder mantener una comunicación más cercana con cada usuario, conociendo con quien interactúan y sus necesidades.

7. ANÁLISIS Y DISEÑO DEL SISTEMA MULTIAGENTE

Para el análisis y diseño del sistema multiagente se utilizó la metodología GAIA y diagramas AUML [2].
Como se define en GAIA, la construcción de un sistema multiagente es un proceso de diseño organizacional, en donde se definen roles y el entorno en donde reside el sistema.

[1] EST: Expressed Sequence Tag: etiqueta de secuencia expresada

7.1 Arquitectura del Sistema Asistente
En la figura 1 se ilustra la arquitectura del sistema asistente y los roles identificados en el mismo.

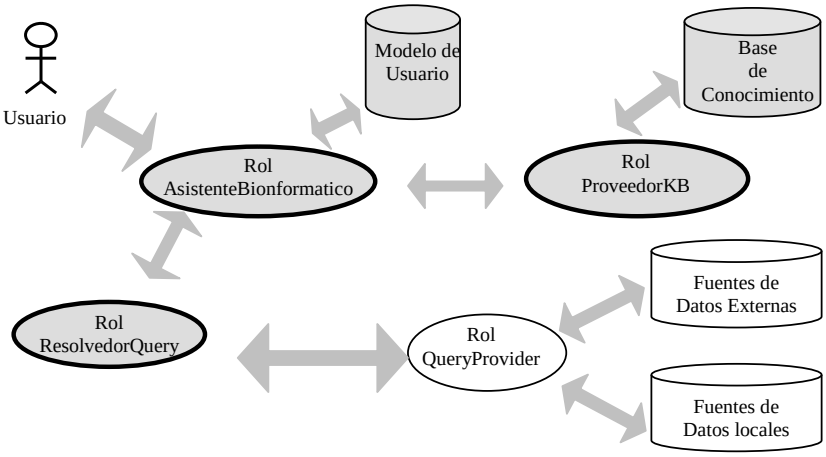


Figura 1: Arquitectura del subsistema asistente

Cuando un usuario accede al sistema a través de la autenticación o de registrarse por primera vez, es atendido por el rol AsistenteBioinformatico, el cual le provee asistencia para ver sus últimas consultas y ver consultas similares de otros usuarios.
Cuando el usuario quiere realizar una consulta, el AsistenteBioinformatico pide al ProveedorKB las consultas posibles a realizar, y luego de que el usuario selecciona la consulta el AsistenteBionformatico despliega los valores que el usuario ingreso como parámetros de dicha consulta la última vez que la realizo.
Para resolver la consulta del usuario el rol AsistenteBioinformatico le pide al ProveedorKB la secuencia de pasos necesaria para resolver la misma. Luego el AsistenteBioinformatico delegará la ejecución de la consulta en el rol ResolvedorQuery, el cual recibe la consulta del usuario, sus parámetros y la secuencia de pasos a realizar. El ResolvedorQuery va delegando la ejecución de cada paso del proceso al rol QueryProvider, para que luego el AsistenteBioinformatico despliegue los resultados de la consulta y los resultados intermedios.
El AsistenteBioinformatico registra la consulta y sus parámetros con el fin de utilizar esta información en el futuro, para proveer al usuario de asistencia.
Si el usuario desea ver el proceso de consulta para alguna de las consultas, el rol AsistenteBioinformatico recibe la petición y solicita el proceso de la consulta seleccionada al rol ProveedorKB.
Por último, el usuario tiene la posibilidad de evaluar el resultado de la consulta. Dicha evaluación es almacenada por el AsistenteBioinformatico y utilizada para mejorar la resolución de próximas consultas, como forma de aprendizaje. Existen dos formas de realizar la evaluación de la consulta: evaluar el resultado final o evaluar los resultados intermedios. El último método proporciona información mas precisa para la tarea de aprendizaje.

7.2 Modelo de Entorno
El entorno en el cual se ejecuta el sistema asistente se compone de los siguientes recursos:

- Interfaz de usuario, de donde se perciben sus acciones y donde se despliegan resultados y sugerencias u otras salidas del sistema
- Modelo de usuario, base de datos donde se almacenan datos referentes a cada usuario del sistema
- Base de conocimiento, base de datos que contiene la información necesaria para que el sistema sepa como resolver una consulta
- Fuentes de datos locales y externas de donde se obtiene la información para obtener los resultados de las consultas del usuario

7.3 Roles Preliminares
Se identificaron los siguientes roles preliminares en el sistema asistente de usuario:

- AsistenteBioinformático, el cual actúa como asistente de usuario, percibiendo las acciones del mismo para poder brindar al usuario de atención personalizada, a través del mantenimiento de un modelo de usuario. Este modelo le permite informar al usuario de sus últimas consultas realizadas, de sus valores ingresados como parámetros, y de las consultas similares de otros usuarios.
- ProveedorKB, el cual es responsable de mantener la base de conocimiento, conocer las diferentes consultas que se pueden realizar en el sistema, y obtener el proceso apropiado para la resolución de una consulta a través de la consulta en la base de conocimiento.
- ResolvedorQuery**,** el cual es responsable de gestionar la resolución de la consulta del usuario, controlando la ejecución de los pasos que componen el proceso de consulta a utilizar para resolverla, y delegar la ejecución de cada paso de una consulta al rol QueryProvider.
- QueryProvider**,** es el rol capaz de resolver los pasos del proceso de consulta, accediendo a fuentes de datos locales y/o externas.

7.4 Modelo de Interacción

Las interacciones entre los roles en el sistema ocurren entre el AsistenteBioinformatico y el ProveedorKB, entre el AsistenteBioinformatico y el ResolvedorQuery, y entre el RevolvedorQuery y el QueryProvider.

La interacción entre los roles AsistenteBioinformatico y Proveedor KB, surge de la necesidad del AsistenteBioinformatico de solicitar servicios al ProveedorKB, para acceder a datos de la base de conocimiento del sistema.

La interacción entre los roles AsistenteBioinformatico y ResolvedorQuery, surge de la necesidad del AsistenteBioinformatico de delegar en el ResolvedorQuery la resolución de una consulta de usuario. El AsistenteBioinformatico percibe los parámetros que el usuario ingresa para la consulta a resolver, y solicita al ProveedorKB el proceso de consulta a utilizar para resolverla. Con estos datos solicita al ResolvedorQuery que se encargue de resolver la secuencia de pasos que componen el proceso de consulta correspondiente.

La interacción entre los roles ResolvedorQuery y el QueryProvider representa los servicios que el ResolvedorQuery solicita al QueryProvider, para resolver cada uno de los pasos del proceso de consulta.

7.5 Modelo de Agentes

Se decidió que cada uno de los roles se corresponda a un clase agente, excepto QueryProvider, el cual es implementado con dos agentes distintos, uno para acceso a bases externas, QueryProviderExterno, y otro para el acceso a la base de datos local, QueryProviderAgentLocal.

Con respecto a las instancias de cada agente, los agentes ResolvedorQueryAgent y AsistenteBioinfoAgent tendrán solo una instancia por usuario, mientras que los agentes QueryProviderLocal, QueryProviderExterno y ProveedorKB tendrán N instancias, siendo N un número entero mayor que uno, a definir dependiendo de la carga del sistema y de la capacidad de hardware con que cuente el servidor en el que corra el sistema.

7.6 Diseño de la Base de Conocimiento

La base de conocimiento almacena la información requerida para saber como resolver la consulta del usuario.

El sistema consulta la base de conocimiento para saber las secuencias de pasos que se requieren para resolver cada consulta.

Por cada paso se almacena en que consiste conceptualmente, para poder explicar al usuario, que servicio del sistema multiagente requiere utilizarse para resolver dicho paso, y sus entradas y salidas.

7.7 Diseño del Modelo de Usuario

Para poder proveer al usuario de asistencia personalizada, el modelo de usuario almacena los siguientes datos por cada usuario del sistema:

- Datos propios del usuario, como ser el nombre, apellido, mail e identificador de usuario.
- Las consultas que realizó.
- Parámetros que ingreso en cada consulta.
- Las evaluaciones que realizó sobre los resultados obtenidos.

8. FLUJO DE TRABAJO DEL SUBSISTEMA ASISTENTE

Cuando un usuario accede al sistema a través de la autenticación o registrándose por primera vez (creando una cuenta de usuario), es atendido por el agente AsistenteBioinfoAgent, el cual le provee asistencia a lo largo de su estadía en la aplicación.

En el momento que el usuario desea realizar una consulta, el agente AsistenteBioinfoAgent solicita a un agente KBProveedorAgent (el cual se seleccionará mediante el uso del protocolo Contract Net) las consultas posibles a realizar. Luego de que el usuario selecciona la consulta que desea, el agente AsistenteBioinfoAgent despliega los valores que el usuario ingreso como parámetros para dicha consulta la última vez que la realizo. Si lo desea, el usuario podrá visualizar los parámetros ingresados por otros usuarios para la misma consulta que está efectuando. A tales efectos, el agente AsistenteBioinfoAgent busca en los modelos de otros usuarios consultas similares a la que se desea realizar y extraerá los parámetros que se utilizaron en las mismas.
Una vez ingresados los parámetros de la consulta, para resolver la misma, el agente AsistenteBioinfoAgent le solicita al agente KBProveedorAgent la secuencia de pasos necesaria para resolver la misma. El agente AsistenteBioinfoAgent delega la ejecución de la consulta en el agente ResolvedorQueryAgent, el cual recibe la consulta del usuario, sus parámetros, y la secuencia de pasos a realizar.
El agente ResolvedorQueryAgent, a su vez, transfiere la ejecución de cada paso de la consulta a los agentes QuetyProviderAgentExterno ó QueryProviderAgentLocal.
Finalmente, el agente AsistenteBioinfoAgent despliega los resultados de la consulta y registra los parámetros de la misma con el fin de utilizar esta información en el futuro, para proveer al usuario de asistencia.
El usuario tiene la posibilidad de evaluar el resultado de la consulta de forma global, o de forma individual, evaluando cada paso del proceso de resolución de la consulta. Dicha evaluación es almacenada por el agente AsistenteBioinfoAgent y utilizada para mejorar la resolución de próximas consultas, aprendiendo de las preferencias del usuario y adaptándose a las mismas.
De forma adicional, el usuario puede consultar el proceso de resolución de consulta para las consultas del sistema. Ante tal petición, el agente AsistenteBioinfoAgent solicita el proceso de la consulta al agente KBProveedorAgent.
Por último, al usuario podrá en consultar el historial de consultas que ha realizado en el sistema. Dicha consulta será resuelta por el agente AsistenteBioinfoAgent.

9. Aspectos Claves del Subsistema Asistente

Existen cuatro pilares fundamentales sobre los cuales se basa el sistema implementado: el primero es el concepto de sistema multiagente; el segundo lo constituyen los agentes asistentes de usuario; el tercero es el aprendizaje del cual están dotados los agentes; y por último las ontologías como mecanismo de representación de conocimiento e intercambio de información.
A continuación se detallará cada uno de los mencionados aspectos y su vinculación con el sistema implementado.

9.1 Sistema Multiagente

Un sistema multiagente es un sistema compuesto por varios agentes que interactúan entre sí, de forma que juntos permiten alcanzar la funcionalidad deseada [3], [4], [5]. En el subsistema propuesto, el objetivo final de poder resolver una consulta bioinformática de acuerdo a los requerimientos planteados, se logra a través de las interacciones entre los diferentes agentes que coexisten en el subsistema: AsistenteBioinfoAgent, KBProveedorAgent, ResolvedorQueryAgent, QueryProviderAgentLocal, QueryProviderAgentExterno.

9.2 Agentes Asistentes de Usuario

Los agentes asistentes de usuario son agentes que asiste activamente a un usuario al operar una interfaz interactiva. Coopera de forma estrecha con el usuario para que éste alcance su objetivo, permitiendo que el usuario aumente su rendimiento y grado de satisfacción, con respecto a la utilización de la aplicación.
Un agente asistente está situado en un entorno, y debe percibir y actuar sobre el mismo. La percepción es lo que le permite recibir las peticiones de los usuarios, de las cuales también aprende. Las acciones del agente asistente constituyen la ayuda que le brinda al usuario a partir de sus peticiones.
Con los agentes asistentes se busca obtener un modelo avanzado de interacción que mejore la comunicación entre el usuario y el sistema, de tal forma que sea el sistema quien aprenda del usuario y no el usuario quien aprenda del sistema. [6], [7], [8], [9], [10], [11], [12].
En el sistema implementado, el agente AsistenteBioinfoAgent es un agente asistente de usuario, el cual utiliza un modelo de usuario para individualizar las características de cada usuario, almacenando sus preferencias y aprendiendo de las mismas, de tal forma de adaptar el sistema al usuario.

9.3 Aprendizaje

Como se define en [5], una de las propiedades de los agentes es su adaptabilidad, lo cual supone que el agente es capaz de actualizar su conocimiento a partir del aprendizaje, aprendiendo de las percepciones que recibe del entorno y de su comportamiento.
En el subsistema asistente se puede encontrar dos mecanismos de aprendizaje:

9.3.1 Aprendizaje a través de la evaluación explícita realizada por los usuarios sobre el resultado de una consulta.
El elemento que le permite saber al sistema si resolvió correctamente una consulta es la evaluación de los usuarios sobre los resultados de las mismas.
Existen dos variantes sobre cómo resolver una consulta: el proceso de consulta utilizado (dado que una misma consulta puede tener asociada diferentes pasos intermedios para ser resuelta) y las fuentes de datos utilizadas para obtener los datos en cada paso del proceso de consulta (las cuales pueden ser locales o externas). Ambas variantes son plausibles de modificación con el objetivo de conseguir mejores resultados para las consultas realizadas.
Para poder relacionar astas dos variantes con la evaluación del usuario, cuando un usuario evalúa los resultados de una consulta, se registra también junto a la evaluación, el proceso de consulta utilizado y qué fuente de datos fue utilizada para resolver cada paso de dicho proceso (QueryProviderAgentLocal para acceso al repositorio local, ó QueryProviderAgentExterno para el acceso a repositorios externos).
Utilizando estos datos, a la hora de resolver una consulta, el sistema es capaz de:

- Seleccionar el proceso de consulta a utilizar, que tenga el mejor promedio de los puntajes obtenidos en las evaluaciones de los usuarios.
- Para cada paso del proceso de consulta, seleccionar la fuente de datos (QueryProviderAgentLocal ó QueryProviderAgentExterno) que tenga el mejor promedio para resolver dicho paso.

De esta manera, el proceso de consulta y la fuente de datos utilizada para resolver cada paso, son los que según los usuarios proveen mejores resultados.

9.3.2 Aprendizaje a través de las observaciones sobre el ambiente.
Los agentes KBProveedorAgent están dotados de cierto mecanismo de aprendizaje que les permiten reproducirse a partir de observaciones del ambiente en el que se ejecutan.
Para evitar la sobrecarga de la ejecución de tareas, estos agentes tienen un máximo de mensajes que pueden estar en cola. Si la cola está llena y se recibe un nuevo mensaje, el mismo será totalmente ignorado. Cada vez que un agente realiza una cotización para resolver una operación sobre la base de conocimiento, se fija en su cola de mensajes la cantidad de mensajes que están pendientes. Si dicha cantidad es superior al 80% de la capacidad máxima de la cola (resta un 20% para que la cola del agente se llene), el agente crea a un nuevo agente KBProveedorAgent en la plataforma.
De esta forma, se realiza un balanceo de carga inteligente, que permite la escalabilidad automática de la aplicación al aumentar el número de usuarios concurrentes que la utilizan.

9.4 Ontologías.
Como se establece en [13], la adquisición de conocimiento se basa en la conceptualización del mismo, lo que implica poseer una visión simplificada y abstracta del dominio que se conoce.
Como se define en [14], una ontología es la descripción formal y explícita de conceptos, dentro de un dominio especificado. Por lo tanto, encontramos en las ontologías el mecanismo buscado para conceptuar el conocimiento.
La especificación y el uso de una ontología para el intercambio de información no solo tienen asociada la posibilidad de la adquisición de conocimiento, sino que define un vocabulario formal para comunicar diferentes entidades que utilizan la misma ontología. Esto establece un compromiso ontológico, a través del cual todas las entidades que manejan una misma ontología se comprometen a utilizar un vocabulario compartido de una forma coherente. Este compromiso ontológico garantiza la consistencia de la información cuando la misma es intercambiada entre diferentes entidades que comparten una misma ontología.
En el sistema se usan dos ontologías:

- AsistenteOntology, la cual conceptualiza el conocimiento relacionado con la asistencia a usuarios. Los agentes AsistenteBioinfoAgent y KBProveedorAgent utilizan la mencionada ontología para el intercambio de mensajes.
- BioinfoOntology, la cual conceptualiza el conocimiento relacionado con la resolución de consultas bioinformáticas. Los agentes ResolvedorQueryAgent, QueryProviderAgentLocal y QueryProviderAgentExterno utilizan la mencionada ontología para el intercambio de mensajes.

10. IMPLEMENTACIÓN
El sistema fue implementado utilizando las librerías de Jade 3.2, lenguaje Java 1.4, tecnologías Web (JSPs, Servlets, Struts), Servidor Web Apache Tomcat 5.0, Eclipse 3.0 como IDE y motor de base de datos SQL Server 7.0.

11. RESULTADOS OBTENIDOS
Como se mencionó anteriormente, los biólogos actualmente deben aprender a utilizar distintos programas y herramientas para resolver una consulta. Estas aplicaciones no son, en general, fáciles de usar, o amigables, y requieren de

conocimiento bioinformático; a su vez, éstas herramientas presentan la complejidad adicional de requerir que el usuario posea refinados conocimientos biológicos, de tal forma que su uso queda restringido al campo de los expertos en la materia

Por lo tanto una de las características deseables es proveer a usuarios ajenos a la bioinformática un sistema fácil de utilizar, que les permita utilizar la información biológica existente, a través de una interfaz simple de manipular, que no requiera de conocimientos de herramientas bioinformáticas.

Las métricas que se utilizaron para medir la facilidad de uso de la solución a construir, comparando con la solución actual, fueron las siguientes:

- Número de interfaces necesarias a conocer por el usuario en contraposición con las interfaces con las que se encontraba en contacto tradicionalmente
- Cantidad de pasos (interacciones) a realizar por el usuario en contraposición con los pasos realizados tradicionalmente, definiendo un paso como una interacción del usuario con una herramienta
- Cantidad de datos a ingresar por el usuario para resolver el requerimiento en contraposición con los ingresados tradicionalmente
- Cantidad de herramientas de consulta y alineamiento de secuencias necesarias a conocer por el usuario en contraposición con las realizadas tradicionalmente
- Cantidad de tipos de usuarios que utilizan el sistema con facilidad en contraposición con la cantidad de tipos de usuario que lo utilizaban tradicionalmente (en este sentido cabe destacar que se validó empíricamente que usuarios del área de la medicina no pudieron ejecutar el proceso tradicional efectuado por los biólogos para la resolución de consultas debido a la complejidad asociada del mismo). Si bien esta métrica puede resultar un tanto subjetiva, uno de los pilares fundamentales que impulsan la construcción del presente trabajo, es la posibilidad de permitir a todo tipo de usuarios con conocimientos básicos, poder utilizar una herramienta bioinformática para resolver consultas. Cada tipo de usuario se define en base a los conocimientos informáticos y biológicos que posee, definiéndose la siguiente matriz que determina los diferentes tipos de usuarios considerados:

Conocimiento biológico → / Conocimiento informático ↓	Avanzado	Corriente
Avanzado	1	2
Corrieonte	3	4

Cuadro 1: Clasificación de los diferentes tipos de usuarios

- Número de facilidades de asistencia al usuario provistas

11.2 Obtención de Métricas

Se pidió a los usuarios potenciales del sistema resolver la consulta definida, a través del sistema multiagente desarrollado, y también siguiendo la secuencia de pasos que realizan habitualmente [15], [16].

El siguiente cuadro muestra el valor para cada una de las métricas obtenidas para ambas soluciones, la actual y la solución construida con tecnología multiagente.

Métrica	**Solución Actual**	**Solución Multiagente**
Número de interfaces necesarias a conocer	13 interfaces	5 interfaces
Cantidad de interacciones a realizar	44 interacciones	2 interacciones
Cantidad de datos a ingresar	30 datos	2 datos
Cantidad de herramientas bioinformáticas a conocer	3 herramientas	0 herramientas
Tipos de usuarios diferentes que pueden utilizar el sistema	2 tipos de usuarios (1,3 en el Cuadro 1)	4 tipos de usuarios (1, 2, 3, 4 en el Cuadro 1)
Cantidad de facilidades de asistencia a usuario ofrecidas	0 facilidades provistas	3 facilidades provistas

Cuadro 2: Métricas obtenidas

En el Anexo A se puede apreciar de forma detallada la forma en que las diferentes normas fueron obtenidas.

12. CONCLUSIONES

A través de las métricas obtenidas, se ha podido apreciar que el sistema desarrollado facilita notablemente la tarea de los investigadores, debido a las siguientes razones:

- El usuario debe ingresar pocos datos para que el sistema pueda resolver su consulta
- El usuario debe conocer pocas interfaces
- El usuario no requiere conocimiento de herramientas bioinformáticas, haciendo posible el uso del sistema a usuarios no expertos en el área de bioinformática.
- El sistema provee al usuario de asistencia, permitiéndole ver sus últimas consultas, consultas similares de otros usuarios y los valores que ingreso como parámetros en la última consulta que realizo.
- El sistema aprende del usuario, adaptándose a sus preferencias a la hora de resolver una consulta.

Teniendo en cuenta las ventajas mencionadas, se concluye que la hipótesis planteada es correcta: *la utilización de agentes inteligentes para el desarrollo de un sistema bioinformático de consultas, permite abstraer al usuario de las herramientas bioinformáticas utilizadas para resolver las consultas y brindarles una interfaz sencilla de utilizar, proveyéndolos de asistencia útil durante la utilización del sistema.*

Otras conclusiones que se desprendieron del trabajo, al aplicar la tecnología de agentes, es que se pudo comprobar la racionalidad e inteligencia de los mismos a través de la implementación de técnicas de negociación y comunicación.

Sin embargo, se estudiaron otras técnicas y arquitecturas que buscan implementar el aprendizaje en los mismos. Su aplicación podría permitir construir un sistema más adaptable al que se ha desarrollado.

Algunas de estas herramientas son Jess [17], el cual es un motor de sistema experto, y Jadex [18], que es un conjunto de clases que permiten construir agentes siguiendo la idea de agentes como entidades inteligentes racionales.

Referencias

[1] Ernesto Ocampo Edye, *"La gestión del conocimiento acumulado en genómica y proteómica: El desafio de la Bioinformática..,* (2004).

[2] Ernesto Ocampo Edye, *"Ingeniería de Software Orientada a Agentes"*, Universidad Pontificia de Salamanca. (2004).

[3] J.M.Corchado. *Teoría de Agentes y Sistemas Multiagentes*. (2004).

[4] Michael Wooldridge y Nicholas Jennings. Intelligent *Agents: Theory and Practice. Knowledge Engineering Review* [Volumen 10 No 2, Junio 1995.]. Universidad de Cambridge. (1995).

[5] Universidad de Oviedo y Pedro Oviedo. *Sistemas Multiagente*. (2001).

[6] Ana Janauskas, Aplicación *de Tecnología de Agentes Asistentes de Usuario para la Integración y Consulta de Repositorios de ESTs*. Estado del Arte. Universidad Católica del Uruguay. (2004).

[7] David Benaderet. *Aplicación de Tecnología de Agentes Asistentes de Usuario para la Integración y Consulta de Repositorios de ESTs*. Estado del Arte. Universidad Católica del Uruguay. (2004).

[8] Antonella Di Steffano, Corrado Santoro, Giuseppe Pappalardo, y Emiliano Tramontana. *Extending Applications using Reflective Assistant Agents*. (2002).

[9] Ana García Serrano, Josefa Z.Hernández, y Paloma Martínez. *El futuro con técnicas de inteligencia artificial. Diálogos en lenguaje natural con asistentes virtuales.* (2002).

[10] Sergio Bermejo y Aniceto Saboya. *Tutores Inteligentes Basados en Asistentes Personales*. (2004).

[11] V.Julian, C.Carrascosa, y M.Rebollo. *Desarrollo de asistentes personales para la gestión de información en Internet*. (2004).

[12] Josefa Z.Hernández. *Agentes Inteligentes y Sistemas Multiagentes*. (2002).

[13] Tom Gruber. *What is an Ontology?* http://www-ksl.stanford.edu/kst/what-is-an-ontology.html. (2004).

[14] Natalya F.Noy y Deborah L.McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. (2004).

[15] Ana Janauskas. *Aplicación de Tecnología de Agentes Asistentes de Usuario para la Integración y Consulta de Repositorios de ESTs.* Construcción de la Norma. (2004).

[16] David Benaderet. *Aplicación de Tecnología de Agentes Asistentes de Usuario para la Integración y Consulta de Repositorios de ESTs*. Construcción de la Norma. (2004).

[17] Ernest J.Friedman-Hill. *Jess, The Expert System Shell for the Java Platform*. (2001).

[18] L.Braubach, W.Lamersdorf, and A.Pokahr. *Jadex: Implementing a BDI infrastructure in Jade*. (2004).

Anexo A: Resultados obtenidos en la evaluación de los sistemas.

Sistema Actual de Realizar la Consulta.

Número de Interfaces:

Sistema o sitio utilizado	**Número de interfaces**
NCBI	2
Wordpad	1
Yahoo	1
SixPack	2
Sitio de Coradhas	1
Clustal	2
HMMER	2
Prosite	2
Total	**13 interfaces**

Cantidad de interacciones:

Sistema o sitio utilizado	**Número de interacciones**
NCBI	2
Wordpad	16
Yahoo	8
SixPack	8
Sitio de Coradhas	2
Clustal	4
HMMER	2
Prosite	2
Total	**44 interacciones**

Cantidad de datos a ingresar:

Sistema o sitio utilizado	**Cantidad de datos ingresados**
NCBI	2
Wordpad	8
Yahoo	4
SixPack	8
Clustal	4
HMMER	2
Prosite	2
Total	**30 datos a ingresar**

Herramientas bioinformáticas:

Herramienta
SixPack
Clustal
HMMER
Total: 3 herramientas

Tipos de usuarios que pueden utilizar el sistema:

Conocimiento biológico → Conocimiento informático ↓	Avanzado	Corriente
Avanzado	1 – Si	2 – No
Corriente	3 – Si	4 – No

Facilidades de asistencia:

No se detectó ningún tipo de asistencia al usuario.

Consulta Realizada a Través del Sistema Multiagente Construido.

Número de Interfaces:

Interfaz
Login
Selección de consulta
Ingreso de parámetros
Resultados
Resultados intermedios
Total: 5 interfaces

Cantidad de interacciones:

Interacción
Autenticación
Resolución consulta
Total: 2 interacciones

Cantidad de datos a ingresar:

Dato ingresado
Consulta a realizar
Especie
Pátogeno
Total: 3 datos

Tipos de usuarios que pueden utilizar el sistema:

Conocimiento biológico → Conocimiento informático ↓	Avanzado	Corriente
Avanzado	1 – Si	2 – Si
Corriente	3 – Si	4 – Si

Facilidades de asistencia:

Facilidad
Historial de consultas del usuario
Historial de consultas similares de otros usuarios
Despliegue de los valores ingresados por el usuario la última vez que realizo la consulta que seleccionó
Total: 3 facilidades

Uma Ferramenta para Converter Exemplos de Aprendizado no Formato Atributo-Valor para o Formato Apropriado ao Uso de Algoritmos de Programação Lógica Indutiva*

Mariza Ferro
Maria Carolina Monard
Laboratório de Inteligência Computacional
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
Av. do Trabalhador São-Carlense, 400 – Centro – Cx. Postal 668
São Carlos – São Paulo – Brasil – CEP 13560-970
{mariza,mcmonard}@icmc.usp.br

Abstract

Typical propositional supervised learning algorithms employ the attribute value representation, which is inadequate for problem-domains that require reasoning about the structure of objects and relations among such objects. On the other hand, Inductive Logic Programming (ILP) is concerned with the development of techniques and tools for relational learning. In this work, we propose a module, named *Kaeru*, to convert data in the attribute-value format to the relational format used by Aleph, a system able to emulate functionalities of several well known ILP systems. The use of *Kaeru* is ilustrated with a simple example. A case study using a real world data set is also described.

Keywords: Artificial Inteligence, Machine Learning, Inductive Logic Programming.

Resumen

Algoritmos de aprendizado proposicional supervisionado geralmente utilizam a representação atributo-valor, a qual é inadequada para problemas nos quais a estrutura dos objetos e as relações entre esses objetos é relevante. Por outro lado, Programação Lógica Indutiva (PLI) esta relacionada com o desenvolvimento de técnicas e ferramentas para aprendizado relacional. Neste trabalho, é proposto um módulo, chamado *Kaeru*, para conversão de dados do formato atributo-valor para o formato relacional utilizado pelo Aleph, um sistema capaz de emular funcionalidades de vários outros sistemas de PLI. O uso do módulo *Kaeru* é ilustrado utilizando-se um exemplo simples. Também é descrito um estudo de caso utilizando uma base de dados real.

Palabras claves: Inteligência Artificial, Aprendizado de Máquina, Programação Lógica Indutiva.

1 Introdução

Atualmente, há um grande interesse na extração de conhecimento de grandes volumes de dados, tanto científicos como industriais. Esse é o objetivo do processo de mineração de dados, que utiliza, entre outros, sistemas de Aprendizado de Máquina (AM) para descobrir padrões e conhecimento útil nos dados.

Ao solucionar problemas com o uso do computador, é importante especificar como traduzí-los em termos computacionais. No caso de AM, isso significa especificar como representar objetos (exemplos), conceitos (hipóteses) induzidos e conhecimento do domínio. O formato atributo-valor, ou proposicional, é a linguagem de representação de objetos mais freqüentemente utilizada em AM, embora sua baixa capacidade de expressão impeça a representação de relações entre objetos ou entre seus componentes. Assim, aspectos relevantes dos objetos, que de alguma maneira poderiam caracterizar o conceito que está sendo aprendido, não podem ser representados utilizando o formato atributo-valor. Essas limitações comprometem a aplicabilidade de

*Trabalho realizado com o auxílio financeiro da CAPES.

algoritmos proposicionais em domínios nos quais a estrutura interna dos objetos de estudo é de grande importância, tais como química, linguagem natural e medicina, entre outros. Outra linguagem de representação mais poderosa que pode ser utilizada na descrição de objetos e conceitos é a linguagem relacional. Os sistemas de aprendizado que utilizam esse tipo de linguagem de representação são chamados de sistemas de aprendizado relacional; nesses sistemas objetos podem ser descritos em termos de seus componentes e relações entre esses componentes. No aprendizado relacional, a linguagem utilizada para descrever exemplos, conhecimento do domínio e descrições dos conceitos é, tipicamente, um subconjunto da Linguagem de Primeira Ordem (LPO) [7], o que permite que os sistemas de aprendizado relacional possam induzir relações ou predicados.

Sistemas de aprendizado que induzem hipóteses na forma de programas lógicos por meio do uso de conhecimento do domínio e uma linguagem de descrição baseada em LPO, ou relacional, são chamados de sistemas de Programação Lógica Indutiva (PLI). Por meio de pesquisas em PLI busca-se superar as limitações da representação proposicional. Esses sistemas possuem uma alta expressividade para representar conceitos e a importante habilidade de representar conhecimento do domínio, o que em geral não acontece com os sistemas de aprendizado proposicional. O conhecimento do domínio utilizado na construção de hipóteses é uma característica importante em PLI pois, quando o conhecimento do domínio é relevante, pode-se melhorar substancialmente os resultados do aprendizado. Entre os diversos sistemas de PLI existentes encontra-se o sistema Aleph [15], que permite simular vários sistemas de PLI, o que motivou seu uso em nossas pesquisas na área de PLI.

Entretanto, existem diversas bases de dados no formato atributo-valor. Para que essas bases possam ser utilizadas pelos sistemas de PLI, elas precisam passar por um processo de transformação que converta esse formato proposicional para um formato relacional equivalente. Neste trabalho é descrito o módulo conversor *Kaeru* [1], cujo objetivo é realizar essa transformação tal que esses dados transformados possam ser diretamente utilizados pelo sistema de PLI Aleph. O uso do módulo *Kaeru* é ilustrado com um exemplo e em um estudo de caso com uma base de dados real da área médica, no qual foi adicionado conhecimento do domínio adquirido utilizando um algoritmo de aprendizado simbólico proposicional.

O restante deste trabalho está organizado da seguinte forma: na Seção 2 são apresentadas definições e características de PLI; na Seção 3 é descrita a ferramenta implementada e um exemplo de uso; na Seção 4 é apresentado o estudo de caso e resultados obtidos com alguns dos experimentos realizados. As conclusões são apresentadas na Seção 5.

2 Programação Lógica Indutiva

PLI é definida como a intersecção de AM indutivo e Programação Lógica [6]. Da lógica computacional, PLI herda seu formalismo representacional, várias técnicas bem estabelecidas e uma profunda base teórica. Do AM indutivo, herda uma abordagem experimental e orientação para aplicações práticas, tais como o desenvolvimento de ferramentas e técnicas para induzir hipóteses a partir de exemplos e sintetizar novos conhecimentos a partir de experiências [9].

PLI se diferencia da maioria dos outros modos de AM pelo uso de uma expressiva linguagem de representação e sua habilidade para utilizar conhecimento do domínio. O problema básico da PLI consiste do aprendizado de definições lógicas a partir de relações, no qual tuplas que pertencem, ou não, à relação alvo, são dadas como exemplos. A partir dos exemplos dados, a PLI induz um programa lógico (definição de predicado) correspondendo a uma teoria que define a relação alvo, em termos de outras relações que são dadas como conhecimento do domínio.

De maneira geral, PLI pode ser descrita a partir de uma teoria de conhecimento do domínio inicial $\mathcal{K}$ e algum conjunto de exemplos $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$, onde $\mathcal{E}^+$ representa os exemplos positivos e $\mathcal{E}^-$ os exemplos negativos do conceito a ser aprendido. O objetivo da PLI consiste em induzir uma hipótese $\boldsymbol{h}$, ou teoria, que junto com $\mathcal{K}$ *explica* os exemplos $\mathcal{E}^+$, ou seja, encontrar uma definição $\boldsymbol{h}$, tal que:

$$\mathcal{K} \wedge \boldsymbol{h} \models \mathcal{E}^+ \text{ e } \mathcal{K} \wedge \boldsymbol{h} \not\models \mathcal{E}^-.$$

É importante observar que o objetivo da PLI é a indução de hipóteses na forma simbólica explícita para que essas hipóteses possam ser facilmente interpretadas pelo usuário/especialista. Assim, o conhecimento induzido pode melhorar o entendimento do problema.

[1] kaeru é uma palavra de origem japonesa que significa converter ou transformar

Entretanto, o espaço das possíveis hipóteses $\boldsymbol{h}$ nesse tipo de aprendizado relacional é muito grande. Assim, para que uma busca nesse espaço seja efetiva é fundamental que ela seja conduzida de maneira sistemática. No caso particular de PLI, para sistematizar a busca no espaço de hipóteses é importante que esse espaço seja estruturado por meio do estabelecimento de uma ordenação de seus elementos. A relação subsunção-θ [9] permite fazer isso.

Com o espaço de hipóteses estruturado pela introdução de uma ordem parcial, buscas nesse espaço tornam-se factíveis e podem ser realizadas sistematicamente, de maneira *bottom-up* (métodos de generalização) ou *top-down* (métodos de especialização). Nos métodos de generalização, as buscas são realizadas utilizando os métodos *generalização menos geral – lgg*[2] [11], *generalização menos geral relativa– rlgg*[3] [11] e *resolução inversa* [10]. Nos métodos de especialização, a busca pode ser realizada utilizando o método de *busca top-down em grafos de refinamento* [14].

Uma variedade de sistemas de PLI têm sido propostos e implementados, entre eles os denominados empíricos, que são sistemas não-interativos de aprendizado não incremental e que aprendem um único predicado. Neste trabalho é usado o sistema empírico de PLI Aleph (A Learning Engine for Proposing Hypotheses) [15], o qual foi desenvolvido com o intuito de ser um protótipo para explorar idéias de PLI. Além disso, Aleph emula funcionalidades de vários sistemas de PLI, o que motivou a sua escolha para o desenvolvimento de nossas pesquisas na área de aprendizado relacional. As execuções de Aleph descritas neste trabalho foram realizadas emulando o sistema de PLI Progol [10].

3 A Ferramenta Proposta

Para aproveitar a expressividade dos sistemas relacionais quando os exemplos estão descritos no formato atributo-valor, é necessário transformar essa descrição proposicional em uma descrição relacional equivalente apropriada. Surge, assim, a necessidade de automatizar, tanto quanto possível, esse processo de transformação. Esse é o objetivo do módulo conversor *Kaeru*, que transforma bases de dados no formato atributo-valor para o formato relacional equivalente, tal que possam ser utilizadas por sistemas de PLI. A seguir é apresentado um exemplo simples que ilustra a utilização e facilidades implementadas no módulo *Kaeru*.

3.1 Exemplo

O conjunto de dados a seguir, foi criado com o objetivo de representar os exemplos que verificam o conceito de `filha` entre o segundo e o terceiro atributos – Tabela 1. Ele consiste de 50 exemplos, 25 deles positivos e os outros 25 negativos, *i.é.*, `classe=sim` ou `classe=nao`, para identificar, respectivamente, se o conceito `filha` é verdadeiro ou falso.

O primeiro atributo `att-id`, identifica o exemplo, o atributo `pessoa1` tem como valores nomes de possíveis filhos (homem ou mulher); o atributo `pessoa2` tem como valores os nomes de possíveis pais ou mães para o atributo `pessoa2`; o atributo `progenitor` pode ser verdadeiro (v) se a `pessoa2` é progenitor da `pessoa1`, ou falso (f) caso contrário.

att-id	pessoa1	pessoa2	sexo-pessoa1	sexo-pessoa2	progenitor	classe
01	ana	maria	fem	fem	v	sim
02	eva	tomas	fem	masc	v	sim
03	carol	tomas	fem	masc	v	sim
04	eva	ana	fem	fem	f	nao
	...	...	...	...	...	
49	laura	beth	fem	fem	f	nao
50	walter	andrea	masc	fem	v	nao

Tabela 1: Conjunto de dados utilizado para o aprendizado do conceito filha.

Esse conjunto de dados no formato atributo-valor, é apropriado para a maioria dos algoritmos de aprendizado supervisionado, entre eles o *See*5 [13], que é uma ferramenta de mineração de dados que induz regras e árvores de decisão para aprendizado proposicional. Neste trabalho, o *See*5 foi utilizado para induzir hipóteses

[2]do inglês, *least general generalization*
[3]do inglês, *relative least general generalization*

descritas por regras de decisão. As regras induzidas pelo $\mathcal{S}ee5$ com o conjunto de dados da Tabela 1, ignorando o `att-id`, já que implicitamente cada linha dessa tabela é reconhecida pelo $\mathcal{S}ee5$ como um exemplo diferente são:

```
Rule 1: (25, lift 1.9)          Rule 2: (17, lift 1.9)        Rule 3: (9, lift 1.8)
  sexo-pessoa1 = fem              progenitor = f                sexo-pessoa1 = masc
  progenitor = v                  ->  classe nao  [0.947]       ->  classe nao  [0.909]
  ->  classe sim  [0.963]
```

A informação da primeira regra, `Rule 1: (25, lift 1.9)`, indica que essa regra cobre 25 exemplos positivos. O valor de `lift` indica o desempenho de cada regra, ou seja, é o resultado da divisão entre o valor da precisão estimada de cada regra pela freqüência relativa da classe predita no conjunto de treinamento. Vale observar que a segunda e terceira regra cobrem, respectivamente, 17 e 9 exemplos negativos, que somadas ultrapassam os 25 exemplos negativos do conjunto de dados. Isso deve-se ao fato de que o $\mathcal{S}ee5$ induz regras sobrepostas, ou seja, mais de uma regra pode cobrir um mesmo exemplo.

Como pode ser observado, essa regra `Rule 1` identifica o conceito de `filha`, caso esse conceito seja já conhecido pelo usuário, mas as regras proposicionais não permitem determinar que essa relação verifica-se entre o atributo `pessoa1` e `pessoa2`.

Uma representação puramente relacional equivalente ao conhecimento implícito na Tabela 1, é mostrada na Tabela 2. Na primeira coluna são apresentados exemplos positivos ($\mathcal{E}^+$) e negativos ($\mathcal{E}^-$) do conceito que se quer aprender (`filha`), indicados, respectivamente, com os símbolos $\oplus$ e $\ominus$; na segunda e terceira coluna há ainda algum conhecimento do domínio declarado de forma extensional, ou seja, contém apenas fatos instanciados; na quarta coluna é declarado conhecimento do domínio de forma intensional. O conhecimento do domínio intensional é muito poderoso e é aceito por alguns sistemas de PLI, entre eles o Aleph.

Exemplos de Treinamento	Conhecimento do Domínio		
$\oplus filha(ana, maria).$	$mae(maria, ana).$	$mulher(ana).$	$progenitor(Pessoa1, Pessoa2) \leftarrow$
$\oplus filha(eva, tomas).$	$mae(ana, tomas).$	$mulher(maria).$	$mae(Pessoa1, Pessoa2).$
...			
$\ominus filha(tomas, ana).$	$pai(tomas, eva).$	$mulher(eva).$	$progenitor(Pessoa1, Pessoa2) \leftarrow$
$\ominus filha(eva, ana).$	$pai(tomas, carol).$	$mulher(carol).$	$pai(Pessoa1, Pessoa2).$
...		$homem(tomas).$	

Tabela 2: Exemplos de treinamento e conhecimento do domínio para aprendizado relacional.

A partir desses exemplos e do conhecimento do domínio fornecido tanto de forma intensional como extensional, o sistema de PLI Aleph pode induzir a seguinte hipótese, utilizando somente `mulher/2` e `progenitor/2` no corpo da regra:

```
filha(Pessoa1,Pessoa2):-mulher(Pessoa1),progenitor(Pessoa2,Pessoa1).
```

a qual indica que a pessoa `Pessoa1` é filha da pessoa `Pessoa2` se a pessoa `Pessoa1` é mulher e se a pessoa `Pessoa2` é progenitor de `Pessoa1`. É importante observar que utilizando essa representação relacional é possível verificar que o conceito de `filha` está relacionado aos atributos `pessoa1` e `pessoa2`, o que não é possível no aprendizado proposicional.

Entretanto, para se utilizar o sistema de PLI Aleph para o aprendizado relacional, essa transformação necessitaria ser feita manualmente para todos os exemplos da Tabela 1. O módulo $\mathcal{K}aeru$, explicado a seguir, ajuda a automatizar essa transformação facilitando esse processo.

3.2 O Módulo Conversor $\mathcal{K}aeru$

O módulo conversor $\mathcal{K}aeru$ é um dos módulos de um ambiente de PLI em desenvolvimento, o qual será integrado ao projeto DISCOVER do Laboratório de Inteligência Computacional (LABIC) do ICMC-USP. O projeto DISCOVER tem como objetivo apoiar todas as etapas do processo de descoberta de conhecimento, *i.é.*, pré-processamento de conjuntos de dados, extração de conhecimento e pós-processamento do conhecimento extraído, e pode ser entendido como um conjunto de métodos para manipular dados e conhecimento por meio de sintaxes padrão para a representação de dados e do conhecimento induzido por diferentes algoritmos de aprendizado [1, 2, 12]. O DISCOVER possui diversas bibliotecas que oferecem um conjunto de funcionalidades

básicas para tais manipulações. O módulo conversor *Kaeru* utiliza as facilidades da biblioteca de classes DOL (uma biblioteca de métodos de pré-processamento de dados) e o ambiente para gerenciamento de experimentos SNIFFER PLI para estimar o erro, usando *k-fold cross-validation*, da hipótese induzida ou criando conjuntos específicos de treinamento e teste para executar Aleph. A Figura 1 mostra como esses ambientes, biblioteca e módulos estão integrados no ambiente PLI. As entradas e saídas do módulo *Kaeru* são explicadas a seguir utilizando o conjunto de dados no formato atributo-valor descrito na Tabela 1.

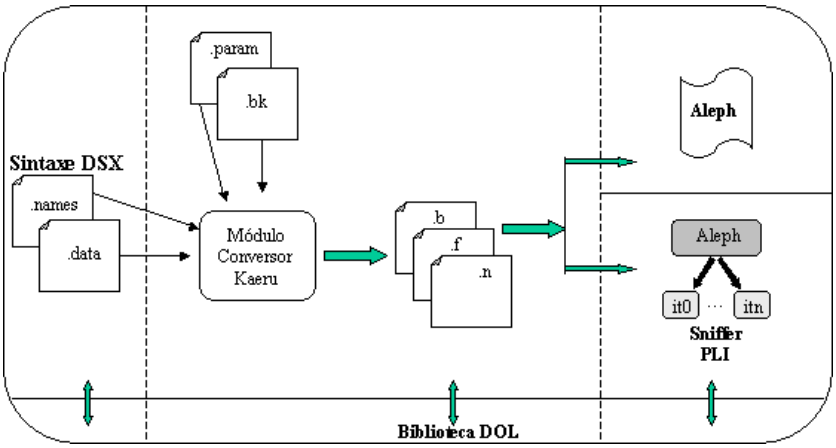


Figura 1: Ambiente de PLI: interação entre o módulo conversor *Kaeru*, o Aleph, a biblioteca DOL e o Ambiente SNIFFER PLI.

Como mencionado, o objetivo do módulo conversor *Kaeru* [4] é automatizar a transformação de dados no formato atributo-valor na sintaxe padrão do DISCOVER para o formato relacional. Além disso, o *Kaeru* também é responsável pela construção das diretivas necessárias para utilizar e executar o sistema Aleph.

A entrada do módulo conversor *Kaeru* consiste dos arquivos `.names` e `.data`, que descrevem os exemplos no formato atributo-valor no formato DSX do DISCOVER [1]. Além desses dois arquivos, também são necessários mais dois arquivos de entrada com extensão `.param` e `.bk`. O arquivo com extensão `.param` contém a declaração dos parâmetros a serem utilizados pelo Aleph, enquanto que o arquivo com extensão `.bk` contém declarações, em um formato padrão específico para o conversor, a serem utilizadas pelo módulo conversor *Kaeru*. Os quatro arquivos de entrada devem ter o mesmo nome, se diferenciando apenas pela sua extensão. Utilizando a informação desses quatro arquivos de entrada, são construídos os três arquivos necessários para a execução do sistema Aleph (os arquivos com extensão `.b`, `.f` e `.n`).

3.2.1 Formato dos Arquivos de Entrada

Dado um conjunto de exemplos no formato atributo-valor, o arquivo com extensão `.names` é utilizado para declarar os atributos, enquanto os valores que esses atributos assumem para esse conjunto de exemplos são declarados no arquivo com extensão `.data`. Para o conjunto de dados da Tabela 1, o conteúdo dos arquivos `.names` e `.data` na sintaxe padrão do DISCOVER é apresentado a seguir

```
%Arquivo .names
sim, nao.

pessoa1: ana, eva,...,walter.
pessoa2: maria, tomas,..., beth.
sexo-pessoa1: fem, masc.
sexo-pessoa2: fem, masc.
progenitor: f,v.
```

```
%Arquivo .data
ana,maria,fem,fem,v,sim
eva,tomas,fem,masc,v,sim
carol,tomas,fem,masc,v,sim
eva,ana,fem,fem,f,nao
          ...
laura,beth,fem,fem,f,nao
walter,andrea,masc,fem,v,nao
```

A primeira linha do arquivo `.names` contem os possíveis valores da classe. Após, encontram-se definidos os atributos, na mesma ordem que aparecem na tabela atributo-valor, junto com os valores desses atributos, caso eles sejam nominais. No arquivo `.data` devem ser definidos os valores dos atributos, na ordem especificada no arquivo `.names`, separados por ','.

A forma mais simples utilizada para definir a informação contida nos outros dois arquivos (`.param` e `.bk`) é apresentado a seguir. Maiores detalhes sobre o possível conteúdo desses arquivos podem ser encontrados em [4].

- Arquivo `.bk`.

```
head{}
body{}
positive{sim}
negative{nao}
```

  Este arquivo contém declarações utilizadas pelo módulo conversor *Kaeru*, as quais são necessárias para a criação do arquivo `.b`.

  O primeiro parâmetro que deve ser declarado no arquivo `.bk` é o `head`. Este parâmetro define a cabeça do predicado (predicado alvo) a ser induzido pelo sistema, ou seja, qual é o conceito que se deseja aprender. No sistema de PLI Aleph, assim como na maioria dos sistemas de PLI, o predicado alvo é definido pelo parâmetro `modeh`. A declaração `head` indica ao conversor qual o predicado alvo do `modeh` para o conjunto de dados apresentado nos arquivos de entrada `.names` e `.data`. Caso não seja fornecido nenhum parâmetro, como no exemplo apresentado, o módulo conversor cria um predicado padrão e que tem sempre aridade 2, `class/2`. No caso do usuário declarar parâmetros para `head`, ele pode definir qual é o predicado alvo desejado. Nesse caso, o usuário pode também definir a aridade do predicado.

  O segundo parâmetro que deve ser declarado no arquivo `bk` é `body`. Ele define os predicados que podem aparecer no corpo das regras induzidas por Aleph. Nos sistemas de PLI, esses predicados são definidos pelo parâmetro `modeb`. Caso não seja fornecido nenhum parâmetro para `body`, como no exemplo apresentado, o conversor cria, por *default*, tantos `modeb` quanto atributos (exceto o atributo `class` e `att_id`) tenha o conjunto de exemplos considerado (arquivo `.names`). O nome de cada um desses atributos é usado para nomear um predicado de aridade 2. No caso do usuário definir os parâmetros de `body`, ele pode definir qualquer número de argumentos (aridade do predicado).

  O terceiro e quarto parâmetro declarados são, respectivamente, os parâmetros `positive` e `negative`. O primeiro é para declarar a classe positiva dos exemplos e o segundo a classe negativa. Essa informação é usada pelo conversor para construir, respectivamente, o arquivo de exemplos positivos `.f` e negativos `.n`.

- Arquivo `.param` - Neste arquivo podem ser declarados qualquer um dos parâmetros do conjunto de parâmetros suportados pelo Aleph [4, 15] na sintaxe usada pelo Aleph. Deve ser observado que todos os parâmetros definidos nesse arquivo são interpretados diretamente pelo Aleph. No exemplo considerado, foi declarado a opção das regras induzidas cobrir no mínimo dois exemplos, usando o seguinte parâmetro.

```
:- set(minpos,2).
```

3.2.2 Formato dos Arquivos de Saída

A sintaxe dos dados nos três arquivos de saída segue a sintaxe do sistema de PLI Aleph [15]. Entre os três arquivos de saída gerados pelo módulo conversor *Kaeru*, encontram-se os arquivos com os exemplos positivos (`.f`) e negativos (`.n`). Os dois arquivos apresentados a seguir, possuem o mesmo formato.

```
%Arquivo .f                  %Arquivo .n
class(0,sim).                class(4,nao).
class(1,sim).                class(5,nao).
class(2,sim).                  ...
    ...                      class(49,nao).
class(48,sim).               class(50,nao).
```

O terceiro arquivo `.b` contém os parâmetros que direcionam a execução do Aleph. Os dados que estão no arquivo de conhecimento do domínio (`.b`), também são gerados automaticamente pelo módulo *Kaeru*, utilizando as informações que estão nos arquivos de entrada, e podem ser divididos nas seguintes partes:

Parâmetros do Aleph São as declarações dos parâmetros do Aleph feitas pelo usuário no arquivo de entrada `.param`;

Declaração dos Modos A segunda parte que é criada pelo módulo *Kaeru* são as declarações dos modos que descrevem os predicados que serão utilizados pelo Aleph. Primeiro é feita a criação dos `modeh`, e para isso o sistema utiliza as informações declaradas no parâmetro `head` do arquivo `.bk`; em seguida o sistema cria todos os `modeb`, utilizando a informação do parâmetro `body` também do arquivo `.bk`;

Determinations A terceira parte é a criação das declarações dos determinations. Esse parâmetro descreve para o Aleph todos os predicados que ele poderá utilizar no processo de indução. Os determinations são criados utilizando a informação dos parâmetros `head` e `body` do arquivo `.bk`.

Declaração dos Tipos A declaração dos tipos é feita pelo módulo *Kaeru* utilizando-se as informações disponíveis no arquivo `.names`. O sistema converte a declaração dos tipos que estão na sintaxe DSX para o formato aceito pela sintaxe do sistema Aleph. São declarados automaticamente pelo sistema todos os atributos que foram utilizados nos predicados e seus respectivos tipos.

Conhecimento do Domínio A última parte criada pelo módulo *Kaeru* é a inclusão do conhecimento do domínio, que é necessário para que o sistema Aleph possa induzir conhecimento utilizando bases de dados originalmente proposicionais. Para isso o módulo *Kaeru* cria, para cada atributo fornecido no arquivo `.names` (exceto para o atributo `class` e `att_id`) uma cláusula Prolog que represente os dados que estão no formato atributo-valor.

A seguir é mostrado um exemplo do arquivo `.b`, criado automaticamente pelo módulo *Kaeru* para o conjunto de dados utilizado para exemplificar o *Kaeru*.

```
% Parâmetros do Aleph %
:- set(minpos,2).

% Declaração dos Modos %
:- modeh(1,classe(+att_id,+att_class)).
:- modeb(*,pessoa1(+att_id,+pessoa1)).
:- modeb(*,pessoa2(+att_id,+pessoa2)).
:- modeb(*,sexo-pessoa1(+att_id,#sexo-pessoa1)).
:- modeb(*,sexo-pessoa2(+att_id,#sexo-pessoa2)).
:- modeb(*,progenitor(+att_id,#progenitor)).

% Determinations %
:- determination(class/2,pessoa1/2).
:- determination(class/2,pessoa2/2).
:- determination(class/2,sexo-pessoa1/2).
:- determination(class/2,sexo-pessoa2/2).
:- determination(class/2,progenitor/2).

% Declaração dos Tipos %
att_id(integer).
pessoa1(ana).       pessoa1(tomas).    ...     pessoa1(walter).
pessoa2(maria).     pessoa2(tomas).     ...    pessoa2(ana).
sexo-pessoa1(fem).          sexo-pessoa1(masc).
sexo-pessoa2(fem).          sexo-pessoa2(masc).
progenitor(f,v).            progenitor(f,v).
att_class(sim).             att_class(nao).

% Conhecimento do Domínio %
pessoa1(Att_id,Pessoa1):-
        example(Att_id,Pessoa1,_,_,_,_,_,_).
pessoa2(Att_id,Pessoa2):-
        example(Att_id,_,Pessoa2,_,_,_,_,_).
sexo-pessoa1(Att_id,Sexo_pessoa1):-
        example(Att_id,_,_,Sexo-pessoa1,_,_,_,_).
sexo-pessoa2(Att_id,Sexo-pessoa2):-
        example(Att_id,_,_,_,Sexo-pessoa2,_,_,_).
progenitor(Att_id,Progenitor):-
        example(Att_id,_,_,_,_,Progenitor,_,_).

example(1,ana,maria,fem,fem,f,v,sim).
```

```
example(2,eva,tomas,fem,masc,f,v,sim).
example(3,carol,tomas,fem,masc,f,v,sim).
                ...
example(50,walter,andrea,masc,fem,f,f,nao).
```

Os arquivos `.b`, `.f` e `.n`, foram fornecidos ao sistema de PLI Aleph que induziu a seguinte teoria:

```
[Rule 1] [Pos cover = 25 Neg cover = 0]
    classe(A,sim) :- sexo-pessoa1(A,fem), progenitor(A,v).
```

Esta regra cobre 25 exemplos positivos (`Pos cover = 25`) e nenhum exemplo negativo (`Neg cover = 0`) e indica que o conceito de `filha` é verdadeiro se o sexo de `pessoa1` é feminino (`fem`) e a relação `progenitor` é verdadeira.

Essa representação é a mais simples que pode ser gerada utilizando o módulo *Kaeru*. Porém, o *Kaeru* possui facilidades que possibilitam criar várias outras representações para esse problema. Por exemplo, utilizando os mesmos arquivos `.names`, `.data`, `.param` e mudando a declaração do arquivo `.bk`, como mostrado a seguir

```
head{1,filha(+att_id,+pessoa1,+pessoa2,+att_class).}
body{}
positive{sim}
negative{nao}
```

O arquivo `.b`, é gerado com um novo `modeh`, ou seja, o conceito que se deseja aprender agora (predicado alvo), além de definir quando o conceito de `filha` é verdadeiro (`classe=sim`), também relaciona quais as pessoas que fazem parte desse conceito (`pessoa1` e `pessoa1`).

Além de declarar esse novo predicado alvo também foi incluído, manualmente, como conhecimento do domínio o conceito de `mulher`, apresentado pela terceira declaração de `modeb` no arquivo `.b` apresentado a seguir, e um novo conceito para `progenitor`, apresentado pela quarta declaração de `modeb` no mesmo arquivo. Dessa forma o novo arquivo `.b` utilizado para induzir o conceito de `filha` é apresentado a seguir, porém, mostrando apenas as partes que foram modificadas em relação ao exemplo anterior:

```
:- modeh(1,filha(+att_id,+pessoa1,+pessoa2,+att_class)).
:- modeb(*,sexo_pessoa2(+att_id,#sexo_pessoa2)).
:- modeb(*,mulher(+att_id,+pessoa1)).
:- modeb(*,progenitor(+att_id,+pessoa1,+pessoa2)).

:- determination(filha/4,mulher/2).
:- determination(filha/4,progenitor/3).
:- determination(filha/4,sexo_pessoa2/2).
:- determination(filha/4,progenitor/2).

%Background
mulher(Att_id,PESSOA1):- example(Att_id,PESSOA1,_,fem,_,_,_).
progenitor(Att_id,PESSOA1,PESSOA2):- example(Att_id,PESSOA1,PESSOA2,_,_,v,_).
```

Utilizando esse novo arquivo `.b`, a seguinte teoria é induzida por Aleph:

```
[Rule 1] [Pos cover = 25 Neg cover = 0]
    filha(A,B,C,sim) :- mulher(A,B), progenitor(A,B,C).
```

Essa regra cobre 25 exemplos positivos e indica que o conceito de `filha` é verdadeiro (pessoa `B` filha de pessoa `C`) se a pessoa `B` é mulher e se pessoa `C` é progenitor de `B`, mostrando assim que há um relacionamento entre essas duas pessoas.

4 Estudo de Caso

O estudo de caso foi realizado utilizando um conjunto de dados real da área médica[4] [5]. O formato original desse conjunto de dados é o formato atributo-valor (proposicional), o qual foi transformado para o formato

[4] A base de dados foi fornecida pelo Dr. Sandro C. Esteves, que juntamente com o Dr. Feng Chung Wu foram os especialistas consultados.

relacional equivalente utilizando o módulo conversor *Kaeru*. Após a transformação, foi utilizado para a realização de experimentos com Aleph, *i.é.,* utilizando aprendizado relacional. O objetivo desses experimentos é verificar a existência de conhecimento relacional nesse conjunto de dados, além da utilização de conhecimento proposicional extraído previamente como conhecimento adicional do domínio para o aprendizado relacional.

4.1 Conjunto de Dados

Milhares de crianças nascem a cada ano. Um número cada vez maior desses nascimentos são originados a partir de casais que antes eram considerados inférteis, graças aos grandes avanços conseguidos no campo do diagnóstico e tratamento da infertilidade conjugal. A dificuldade para ter filhos é muito mais comum do que se imagina; de cada seis casais, um terá dificuldade para tê-los. O conjunto de dados do estudo de caso está relacionado com a reprodução humana assistida. Ele consiste de 278 exemplos descritos por 19 atributos que contém os resultados obtidos de um espermograma e um processamento de sêmen de um mesmo paciente.

Em todas as situações onde houver indicação de reprodução assistida uma ou mais análises deve ser realizada no sêmen. O espermograma é um exame laboratorial que determina o número, a movimentação (motilidade) e o formato dos espermatozóides. Além disso, o espermograma verifica se a quantidade de líqüido seminal (volume) é normal, se há sinais de infecção e várias outras medidas. Porém, existe outro exame que avalia com maior precisão a qualidade do sêmen; esse exame é chamado de processamento de sêmen diagnóstico. Entretanto, contrariamente ao espermograma, o processamento de sêmen é bastante caro e pode elevar o custo do exame em aproximadamente 80% em relação à um espermograma.

O processamento de sêmen pode separar os melhores espermatozóides, melhorar o potencial fértil dos espermatozóides e ainda determinar com maior precisão o número de espermatozóides recuperados de boa qualidade. O número de espermatozóides recuperados é um dos critérios que auxilia na escolha da técnica de reprodução assistida mais apropriada para o casal.

Porém, como já citado, o processamento de sêmen, apesar de mais preciso, é bem mais caro que o espermograma. Assim, além do interesse em extrair e avaliar o conhecimento adquirido, um dos interesses do estudo desse tema é tentar predizer qual será a quantidade de espermatozóides com boa motilidade recuperados pelo processamento de sêmen antes mesmo da realização desse exame, a partir de exames menos custosos como o espermograma. Assim, utilizando as medidas que são realizadas durante o processamento de sêmen para a determinação das classes e as medidas realizadas no espermograma são fornecidos como atributos ao algoritmo de indução. Em outras palavras, todos os pacientes nessa base de dados realizaram ambos exames laboratoriais: espermograma (baixo custo) e processamento de sêmen (alto custo). Para cada paciente, a classe é determinada pelo resultado obtido no exame de processamento de sêmen (alto custo), enquanto que os valores dos atributos correspondem às diversas medidas realizadas no espermograma (baixo custo).

Existem três principais técnicas de reprodução assistida: inseminação intra uterina (IUI), fertilização in vitro (FIV) e a injeção intracitoplasmatica de espermatozóides no óvulo (ICSI). Baseado nessas técnicas de reprodução, durante a fase de entrevistas com o especialista realizada em estudos anteriores [8, 3], foram definidas as seguintes classes, onde a variável x representa milhões de espermatozóides por mililitro (ml) recuperados pelo processamento de sêmen:

- Classe = ICSI: $x < 1$;
- Classe = FIV: $1 \leq x < 5$;
- Classe = IUI: $x \geq 5$.

A Tabela 3 mostra o resumo desse conjunto de dados: número de exemplos (#Exemplos); número total de atributos e o número de atributos contínuos e nominais (#Atributos(cont.,nom)); as classes e suas distribuições (Classes e Classes %); erro da classe majoritária (Erro CM).

4.1.1 Descrição dos Experimentos

O conjunto original de dados foi transformado para o formato relacional equivalente utilizando o módulo conversor *Kaeru* e foi submetido ao sistema de PLI Aleph, ajustando o parâmetro do Aleph que define o número mínimo de exemplos que devem ser cobertos por uma regra, o qual foi ajustado para 2 (dois).

# Exemplos	# Atributos	Classes	Classes %	Erro CM
278	19 (14,5)	ICSI	40,5%	59%
		FIV	22,5%	sobre ICSI
		IUI	37,0%	

Tabela 3: Resumo do conjunto de dados processamento de sêmen utilizado nos experimentos.

Foram realizados diversos experimentos, os quais podem ser divididos em duas etapas. Na primeira etapa, os experimentos foram realizados utilizando, como conhecimento adicional, o conhecimento fornecido pelo especialista. Como resultado desses primeiros experimentos, o número de regras induzidas foi alto, sendo que a maioria das regras cobrem apenas dois exemplos. Assim, na segunda etapa, foi introduzido mais conhecimento do domínio e foi utilizado, como conhecimento adicional, o conhecimento extraído desse conjunto de dados utilizando o algoritmo de aprendizado proposicional $\mathcal{S}ee5$, além do conhecimento fornecido pelo especialista já utilizado na etapa anterior. Contrariamente a maioria dos sistemas de PLI o $\mathcal{S}ee5$ consegue manipular bem atributos contínuos, achando bons pontos de corte, os quais podem ser utilizados para incrementar o conhecimento do domínio. A fim de observar o efeito desse conhecimento adicional, foi decidido incorporá-lo em várias fases, induzindo o modelo correspondente a cada fase.

Um exemplo de conhecimento adquirido utilizando $\mathcal{S}ee5$ refere-se a divisão dos valores do atributo `class-a` (que representa a porcentagem de espermatozóides com motilidade grau A) em 4 faixas (baixa, média-baixa, média-alta e alta). Esse conhecimento, definido pelos seguintes predicados, assim como o conhecimento relacionado a divisão dos valores de outros atributos contínuos foi inserido como conhecimento de fundo.

```
e_classea(Att_id,baixa,6):- class_a(Att_id,CLASS_A), menor_igual(CLASS_A,6).
e_classea(Att_id,media_baixa,6):- class_a(Att_id,CLASS_A), maior(CLASS_A,6).
e_classea(Att_id,media_alta,17):- class_a(Att_id,CLASS_A), menor_igual(CLASS_A,17).
e_classea(Att_id,alta,17):- class_a(Att_id,CLASS_A), maior(CLASS_A,17).
```

Diversos experimentos foram realizados inserindo passo a passo o conhecimento de fundo, bem como considerando diferentes classes como classe positiva e as classes restantes como negativas. O conhecimento de fundo refere-se ao conhecimento adicional fornecido pelo especialista, quanto ao conhecimento de divisão de valores de atributos contínuos obtidos com $\mathcal{S}ee5$. A Tabela 4 mostra alguns resultados obtidos em três desses vários experimentos realizados. A coluna Experimento identifica o experimento, em outras palavras, o conhecimento de fundo utilizado para induzir o modelo; Classe Positiva% mostra a classe utilizada como positiva, entre as três classes existentes no conjunto de dados e sua distribuição; No Regras refere-se ao número de regras no modelo induzido; finalmente Erro 10-cv mostra o erro médio do modelo induzido calculado utilizando 10-*fold cross validation*, conjuntamente com a variância.

Experimento	Classe Positiva (%)	No Regras	Erro 10-cv
A	ICSI(40,5%)	20	27,3(7,1)
B	ICSI(40,5%)	21	23,0(7,8)
C	ICSI + FIV (63,0%)	28	32,0(8,6)

Tabela 4: Resultados de alguns experimentos realizados com o conjunto processamento de sêmen.

Deve ser lembrado que nessa base de dados os valores dos atributos correspondem a um exame de laboratório de baixo custo (espermograma) enquanto que a classe é dada por um exame de alto custo (processamento de sêmen). Um dos objetivos é verificar em que casos é possível determinar essa classe, para novos pacientes, considerando os resultados do exame de menos custo. É evidente que nesse tipo de problemas, freqüente na área médica, é difícil conseguir um modelo para classificar bem todos os novos casos não vistos, já que isso significaria que um exame pode ser substituído pelo outro. Se esse for o caso, o médico sempre recomendaria o exame menos custoso. Como pode ser observado na Tabela 4, isso se verifica nesse estudo de caso, já que o erro do classificador induzido é alto. Entretanto, é interessante para o especialista a procura de algumas regras desse modelo, tal que, para alguns casos, seja possível predizer a classe sem

realizar o exame mais custoso. Esse conhecimento pode ser extraído considerando as regras do modelo que cobrem um número considerável de exemplos.

Considerando os modelos induzidos, o número de regras e a cobertura dessas regras, os experimentos A, B e C apresentaram alguns resultados considerados interessantes pelos especialistas. A fim de ilustrar, são apresentadas algumas regras do modelo A que cobrem bastante exemplos positivos (`Pos cover`) e nenhum exemplo negativo (`Neg Cover`):

```
[Rule 1][Pos cover = 22 Neg cover = 0]
 class(A,icsi) :- hora(A,tarde), e_classea(A,media_alta,17),concentracao_esp(A,baixa,29.7).

[Rule 4][Pos cover = 30 Neg cover = 0]
 class(A,icsi) :- class_a(A,0), aha_p(A,baixa,58).

[Rule 7][Pos cover = 18 Neg cover = 0]
 class(A,icsi) :- processamento(A,20), concentracao_esp(A,baixa,29.7),aha_p(A,baixa,58).
```

É interessante notar que o modelo A utiliza ICSI como classe positiva. A técnica ICSI é a mais sofisticada e utilizada quando há poucos espermatozóides por ml e/ou baixa qualidade dos espermatozóides (por exemplo baixa motilidade). Algumas relações que aparecem no corpo das regras chamam a atenção e demonstram coerência no conhecimento induzido. Por exemplo: a relação `hora(tarde)`, que está relacionada com a hora de coleta de sêmen, aparece no corpo de várias regras tanto no modelo A quanto nos modelos B e C. Isso pode ser interpretado de várias maneiras, uma delas é que o ciclo fisiológico influencia no resultado da coleta.

A relação `e_classea(A,media_alta,17)` também aparece em várias regras de todos os modelos. Essa relação é verdadeira quando a motilidade de grau A (melhor motilidade medida) é menor ou igual a 17%, ou a relação `class_a(A,0)` a qual indica que a motilidade de grau A é 0%, ou seja, a quantidade de espermatozóides com boa motilidade é bastante baixa. No modelo A, das três regras que cobrem mais exemplos, a relação `concentracao_esp(A,baixa,29.7)` aparece em duas dessas regras. Essa relação é verdadeira quando a concentração de espermatozóides por ml é menor ou igual a 29.7 (em milhões de espermatozóides), ou seja, uma baixa concentração de espermatozóides. É interessante notar que as regras indicam, por exemplo, que pacientes sem espermatozóides com baixa motilidade e com baixa concentração de espermatozóides por ml precisam ser tratados utilizando a técnica ICSI.

Ainda que as regras aqui apresentadas são regras proposicionais, elas servem de base para o especialista descobrir se existem relações entre os diversos atributos desse conjunto de dados. Esse trabalho deve ser realizado conjuntamente com o especialista, redefinindo tanto a aridade da relação classe (observar que nas regras apresentadas é utilizada a sua forma mais simples com aridade 2) bem como a aridade das relações que compoem o conhecimento de fundo, tal que essas relações agrupem atributos que, segundo o especialista, tem potencial de estarem relacionadas.

Esse trabalho tanto pode ser realizado executando novamente o módulo *Kaeru* especificando a aridade e atributos participantes de cada relação, como também pode ser realizado manualmente definindo novos relacionamentos em função das relações mais simples de aridade 2, que relacionam cada atributo de um exemplo com seu valor e inserindo essas novas relações no corpo das regras por meio das declarações do Aleph `modeh` e `modeb`.

5 Considerações Finais

Neste trabalho é apresentada a ferramenta *Kaeru* que transforma bases de dados no formato atributo-valor para o formato relacional equivalente. Além disso, o módulo *Kaeru* também cria as diretivas necessárias para executar o sistema de PLI Aleph. Além da conversão simples dos conjuntos de dados para o formato relacional, o módulo *Kaeru* também apresenta outras facilidades, tais como a criação de diferentes predicados para o aprendizado, a criação de diferentes arquivos de exemplos positivos e negativos, entre outros, por meio da declaração de parâmetros e comandos simples. A utilização e algumas facilidades implementadas no módulo *Kaeru* foram ilustradas utilizando um exemplo simples.

Neste trabalho também é apresentado um estudo de caso utilizando um conjunto de dados do mundo real no formato atributo-valor, o qual foi transformado para o formato relacional do sistema Aleph utilizando o módulo conversor *Kaeru*. Na fase de escolha do conhecimento do domínio foram definidos alguns predicados sugeridos pelo especialista e, também, foi definido conhecimento adicional do domínio utilizando o conhecimento induzido pelo algoritmo de aprendizado proposicional *See*5.

Vale lembrar que, devido à maneira como essa base de dados foi construída, *i.e.* os valores dos atributos provem de um exame de laboratório de baixo custo, enquanto que as classes provem de um outro exame de

maior custo, a indução de um modelo que classifique com precisão qualquer novo caso é utópica. O objetivo nesse tipo de problema, muito freqüente na área médica, é encontrar um conjunto de regras que classifique bem alguns casos. Assim, novos casos cobertos por essas regras, indicam ao médico que não há necessidade de realizar o exame de alto custo pois é possível classificar esses casos utilizando os resultados do exame de baixo custo realizado por esses pacientes. Os resultados obtidos mostram que a utilização de conhecimento adicional do domínio, obtido a partir de modelos proposicionais, pode auxiliar no aprendizado relacional auxiliando na definição de pontos de cortes apropriados no caso de atributos contínuos, já que a maioria dos algoritmos de PLI não consegue lidar bem com atributos contínuos.

Referências

[1] Gustavo E. A. P. A. Batista. Sintaxe padrão do arquivo de exemplos do projeto Discover, 2001. `http://www.icmc.sc.usp.br/~gbatista/Discover/SintaxePadraoFinal.htm`.

[2] G. E. A. P. A. Batista and M. C. Monard. Descrição da Arquitetura e do Projeto do Ambiente Computacional Discover Learning Environment — DLE. Technical Report 187, ICMC-USP, 2003. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_187.pdf`.

[3] Mariza Ferro, Huei Diana Lee, and Sandro C. Esteves. Intelligent Data Analysis: A Case Study of the Diagnostic Sperm Processing. In *Proceedings ACIS International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA-02)*, pages 116–120, Foz do Iguaçu, Brazil, June 2002.

[4] Mariza Ferro and Maria Carolina Monard. Descrição da Implementação do Módulo Conversor *Kaeru* para Transformar Dados no Formato Atributo-Valor para o Formato Relacional do Sistema de Programação Lógica Indutiva Aleph. Technical Report 234, ICMC-USP, 2004. `ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel\_tec/RT\_234.pdf`.

[5] Mariza Ferro. Aquisição de conhecimento de conjuntos de exemplos no formato atributo valor utilizando aprendizado de máquina relacional, 2004. Dissertação de Mestrado, ICMC-USP, `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-16112004-095938/%publico/Dissertacao.pdf`.

[6] Nada Lavrač and Peter A. Flach. An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic (TOCL)*, 2(4):458–494, 2001.

[7] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.

[8] Huei Diana Lee. Seleção e construção de features relevantes para o aprendizado de máquina, 2000. Dissertação de Mestrado, ICMC-USP.

[9] S. Muggleton and L. De Raedt. Inductive Logic Programming. *Journal of Logic Programming*, 19–20:629–679, 1994.

[10] S. Muggleton. Inverse Entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[11] G. D. Plotkin. A Further Note on Inductive Generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6, New York, 1971. Elsevier.

[12] Ronaldo Cristiano Prati, José Augusto Baranauskas, and Maria Carolina Monard. Uma proposta de unificação da linguagem de representação de conceitos de algoritmos de aprendizado de máquina simbólicos. Technical Report 137, ICMC-USP, 2001. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_137.ps.zip`.

[13] Rulequest-Research. Data Mining Tools See5 and C5.0, 1999. `http://www.rulequest.com/see5-info.htmlem03/1999`.

[14] E. Y. Shapiro. *Algorithmic Program Debugging*. The MIT Press, 1983.

[15] A. Srinivasan. The aleph manual. Technical report, Oxford University, 2000. `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph_to%c.html`.

ϵ-PSO: A Particle Swarm Optimizer with Epsilon-Greedy Action Selection

Jorge I. Barrerra
Universidad Pontificia Bolivariana, Antioquia,
Medellín, Colombia
jorgeb49@yahoo.es

Jorge A. Peña
Universidad Pontificia Bolivariana, Antioquia,
Medellín, Colombia
joralepe@gmail.com

and

Roberto C. Hincapié
Universidad Pontificia Bolivariana, Antioquia,
Medellín, Colombia
roberto@upb.edu.co

Abstract

The necessary balance between global and local search, typical of optimization problems, is similar to that between exploration and exploitation in reinforcement learning problems. In particle swarm optimization (PSO), the inertia weight parameter has been typically used to balance global and local search. In this paper we proposed ϵ-PSO, a PSO algorithm in which each particle in the swarm behaves following an epsilon-greedy stochastic policy. That is, instead of always updating its velocity according to the neighborhood best, the particle best solution already found by another neighbor is sometimes used, thus preventing the swarm from premature convergence. Results of the new algorithm in three well known test functions show a better performance of the proposed algorithm compared to that of standard PSO, especially when damping of particle position is required.

Keywords: Artificial intelligence, Optimization methods, Particle swarm optimization, Reinforcement learning.

1 INTRODUCTION

Particle Swarm Optimization (PSO) is a population-based optimization method first introduced by Kennedy and Eberhart [3], [4]. The algorithm is founded on the social behavior of certain species and evolutionary psychology, which suggests that sociocognitive individuals must be influenced by their past behavior and the success of their neighbors. PSO has become widely used in engineering and computer sciences as a problem-solving method and it is now a competitive alternative to other stochastic optimization methods such as genetic algorithms and evolution strategies.

Several variations to the original PSO algorithm have been proposed in order to prevent the premature convergence and the stagnation generally shown by the original algorithm. Some of these variants can be found in [1], [2], [5], [6] [10], [11], [14], [15], [16]. Some approaches are computationally expensive, while others are cheaper. Note, however, that the list is far from being complete (or even representative).

Here, a variation of the original algorithm that provides it with an elegant mechanism borrowed from the reinforcement learning literature to balance exploration and exploitation is presented. At the best knowledge

of the authors, the proposed extension is an original one, and the resulting algorithm is computationally inexpensive and general enough to open the possibility of being combined, if necessary, with any of the referred approaches in order to obtain an even more powerful method.

The paper is organized as follows: section 2 introduces the original particle swarm optimization. Section 3 reviews some essentials of reinforcement learning (particularly the Q-learning algorithm) and stochastic action selection (particularly ϵ-greedy action selection). In section 4, our ϵ-PSO algorithm is presented. Section 5 and 6 present the experimental settings to test the new algorithm and the results obtained. Finally, section 7 outlines some conclusions and directions for future work.

2 PARTICLE SWARM OPTIMIZATION

In PSO, a D-dimensional search space is explored using a swarm of particles with random position and velocity initialization, seeking to minimize a multivariable function f. Particle i keeps records of its position $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, its velocity $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ and the best position it has found so long $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$. Each particle has two kinds of available information, used to update its position each time step: (a) the knowledge of its particular experience and (b) the knowledge of its neighbors experience. More specifically, the velocity of each particle i is updated taking into account the vector $\mathbf{p}$ with the best fitness in the local neighborhood ($\mathbf{p}_g$, aka *gbest*), and the vector $\mathbf{p}$ of particle i itself ($\mathbf{p}_i$, aka *pbest*). The new velocity is added to the previous position to obtain the particle new position. The dynamics of particle i are given by the following two equations:

$$v_{id} = v_{id} + c_1 \cdot rand() \cdot (p_{id} - x_{id}) + c_2 \cdot Rand() \cdot (p_{gd} - x_{id}) \tag{1}$$

$$x_{id} = x_{id} + v_{id} \tag{2}$$

where c_1 and c_2 are the cognitive and social factors (usually set to 2), $rand()$ and $Rand()$ are two different random functions in the range $[0, 1]$, and $g = \arg\min_i f(\mathbf{p}_i)$. Before updating the particle position according to Eq. 2, it is usual to implement a damping function that restricts the velocity, this function is given by [4]:

$$v_{id} = \begin{cases} v_{max} & \text{if } v_{id} > v_{max} \\ -v_{max} & \text{if } v_{id} < -v_{max} \\ v_{id} & \text{otherwise} \end{cases} \tag{3}$$

A PSO with the velocity update formula given by Eq. 1 is usually slow, because the particles oscillate much before converging near a local optimum. A simple method to prevent this excess of oscillation is to include a parameter called inertia weight $0 \leq w \leq 1$, so that the velocity update formula is now given by:

$$v_{id} = w \cdot v_{id} + c_1 \cdot rand() \cdot (p_{id} - x_{id}) + c_2 \cdot Rand() \cdot (p_{gd} - x_{id}) \tag{4}$$

The inertia weight w has been traditionally used to balance global and local search, [7], [8]. In fact, a value of this parameter close to 1 cause the particles to oscillate wider, thus forcing them to explore more widely, whereas a relatively small value force them to focus their search on a narrower region. The use of inertia weight also guarantees the converegence of the system using a wide range of v_{max}, that means that v_{max} is then not a critial parameter if the inertia weight is used, giving robustness to the system [4]. In [9], a linear decreasing from an initial value of 0.9 and a final value of 0.4, was determined to be a suitable schedule for this parameter, allowing more global search at the beginning and more local search at the end of the run.

3 REINFORCEMENT LEARNING AND ϵ-GREEDY ACTION SELECTION

Reinforcement learning is the problem of learning a *policy*, a mapping between a set of states S and a set of actions A so as to maximize a numerical reward [12]. The agent-environment interaction follows discrete, rather simple dynamics: (a) in time step t the agent finds itself in a given state s_t, (b) it chooses an action a_t according to the current policy, and (c) one time step later, the environment answers with a numerical reward r_{t+1} and a new state s_{t+1} that is presented to the agent.

Q-learning, one of the most known reinforcement learning algorithms, is based on the incremental estimation of a Q table, which stores the quality value of taking each available action a in each single state s in terms of future received reward. It has been proved that the *greedy* policy over Q (the policy that chooses $a_t = \arg\max_a Q(s_t, a)$) converges to the optimal policy [13].

While learning, however, a *stochastic policy*, which not always chooses the action with the maximum Q value for the current state, is usually followed instead of the greedy one. A stochastic policy assigns a probability of being taken different from zero to each action available in a given state. This is done in order to deal with the *exploitation-exploration dilemma*: the agent has to exploit what it already has learned to choose the action with the maximum current value in order to receive as much reward as possible, but it also has to explore other actions that currently seem poor to determine if a lesser Q value is due to a poor estimation and not to an explicit disadvantage of such an action. At the beginning of the learning process, when Q values are poorly estimated by the iterative algorithm, it is convenient to allow a great deal of exploration, whereas at the end it is better to behave more greedily, because the Q values have been estimated and could be near to their real values.

The simplest stochastic policy is the ϵ-greedy policy. The ϵ-greedy policy assigns a probability of $1 - \epsilon$ to the greedy action and a probability of ϵ to the other actions. In the case of a Q-based algorithm, the ϵ-greedy policy assigns to each action in state s_t a probability $\pi(s_t, a)$ of being taken given by:

$$\pi(s_t, a) = \begin{cases} 1 - \epsilon & \text{if } a = \arg\max_a Q(s_t, a) \\ \frac{\epsilon}{|A(s_t)|-1} & \text{otherwise} \end{cases} \tag{5}$$

where $|A(s_t)|$ is the number of available actions in state s_t.

In most applications, ϵ is a decreasing function of the time step t. The overall effect is to encourage exploration at the beginning of the run and exploitation at the end.

4 PARTICLE SWARM OPTIMIZATION WITH ϵ-GREEDY ACTION SELECTION

Dynamics of a particle in a particle swarm are not very dissimilar to that of a simple reinforcement agent: the particle finds itself in a given state (position), takes an action (updates its velocity), and finds itself one step later in other state depending on the previous state and on the taken action. Assuming the stochasticity of the particle's movement (given by the *rand*() and *Rand*() functions of Eq. 1) to be caused by the intrinsic dynamics of the environment, it can be argued that each particle in the swarm follows a deterministic, greedy policy in relation to the vectors $\mathbf{p}$ of its neighborhood. That is, particle i greedily chooses $g = \arg\min_i f(\mathbf{p}_i)$ and then updates its velocity according to Eq. 1.

This paper proposes a more general particle swarm in which the choice of g is not a deterministic greedy policy, but a stochastic ϵ-greedy policy. In other words, each vector $\mathbf{p}_i$ has a probability $\pi(i)$ different from zero of becoming the *gbest* for another particle in the same local neighborhood:

$$\pi(i) = \begin{cases} 1 - \epsilon & \text{if } i = \arg\min_i f(\mathbf{p}_i) \\ \frac{\epsilon}{N-1} & \text{otherwise} \end{cases} \tag{6}$$

where N is the number of particles in the local neighborhood. It is expected that, as in the reinforcement learning case, a large ϵ favors exploration, whereas a small ϵ favors exploitation.

As defined above, ϵ-PSO defines a superset of algorithms with standard PSO as a special case when $\epsilon = 0$.

5 EXPERIMENTAL SETTINGS

Two experiments were devised in order to test the proposed algorithm and to compare it with the standard algorithm. In the first setting, only the velocities of the particles are dampen according to Eq. 3. In the second setting, the positions of the particles are also dampen after the calculation of Eq. 2, according to:

$$x_{id} = \begin{cases} x_{max} & \text{if } x_{id} > x_{max} \\ -x_{max} & \text{if } x_{id} < -x_{max} \\ x_{id} & \text{otherwise} \end{cases} \tag{7}$$

It is worth to note that position damping is included here not in order to improve the algorithm, but rather to take into account the cases where damping of particle position is explicitly required. In practical problems it is frequently needed to restrict the particles to a certain region of the search space because of physical constraints. For instance, if a PSO algorithm is intended to tune a real PID controller, the positions of the particles (which represents the proportional, integral and derivative constants of the controller) can not go beyond the limits of the physical device. Another case in which position damping would be necessary is an embedded implementation of a PSO algorithm, where the precision of the manipulated numbers may be scarce (for instance, an 8-bit architecture).

For the purpose of comparation, three benchmark functions commonly used in the literature are used here in order to test the proposed algorithm.

The function f_1 is the Rosenbrock function:

$$f_1(\vec{x}) = \sum_{d=1}^{D} 100(x_{d+1} - x_d^2)^2 + (x_d - 1)^2 \tag{8}$$

The function f_2 is the generalized Rastrigin function:

$$f_2(\vec{x}) = \sum_{d=1}^{D} x_i^2 + 10[1 - \cos(2\pi x_d)] \tag{9}$$

The function f_3 is the generalized Griewank function:

$$f_3(\vec{x}) = \frac{1}{4000} \sum_{d=1}^{D} x_d^2 - \prod_{d=1}^{D} \cos\left(\frac{x_d}{\sqrt{d}}\right) + 1 \tag{10}$$

The positions of the particles are initialized according to the asymmetric initialization method used in [9]. Table 1 shows the values of v_{max} and x_{max} besides the initialization ranges for each function. These values are also borrowed from [9] for comparison purposes. In the first experiment x_{max} serves only to indicate the maximum velocity v_{max} allowed (note that $x_{max} = v_{max}$). In the second experiment x_{max} is directly used to dampen the position of the particles according to Eq. 7.

Function	Initialization Range	v_{max}	x_{max}
Rosenbrock	$(-100, 100)^D$	100	100
Griewank	$(-600, 600)^D$	10	10
Rastrigin	$(-5.12, 5.12)^D$	600	600

Table 1: Initialization ranges, v_{max} and x_{max} for each function

Again according to [9], dimension sizes of 10, 20 and 30 are tested. The maximum number of iterations G_{max} is set to 1000, 1500 and 2000, corresponding to dimension sizes of 10, 20 and 30, respectively. Populations sizes of 20, 40, 80 and 160 are also used.

A linearly decreasing ϵ starting at ϵ_0 and ending at 0 is used, with $w = 0.4$, $c_1 = 2$ and $c_2 = 2$. Furthermore, the velocity of the particles is dampen according to Eq. 3 and the values of v_{max} given in Table 1.

A star or *gbest* topology is used, so $N = m$, where m is the total number of particles in the swarm. In other words, the whole population is contained in one single neighborhood. A total number of 50 runs were allowed each time.

6 RESULTS AND DISCUSSION

6.1 First experiment

Table 2, Table 3 and Table 4 respectively show the mean fitness values for the Rosenbrock, the Rastrigin and the Griewank function obtained after 50 runs of the proposed algorithmm, with values of ϵ_0 of 0.8, 0.6, 0.4 and 0.2. For comparison purposes, the results reported in [9] for the standard PSO algorithm with a

linear decreasing w from 0.9 to 0.4 (henceforth called SPSO) are also listed. The minimum mean fitness value obtained by any algorithm is emphasized in boldface.

As it can be seen, at least one variant of ϵ-PSO performed better than SPSO for all the tested cases in the Rosenbrock and Griewank function. On the other hand, SPSO performed better than every variant of ϵ-PSO in the Rastrigin function for all the population sizes and dimension sizes greater than 10 (except for a population size of 160 and a dimension size of 20). In regard to ϵ-PSO, an initial value of 0.2 seemed to be the most indicated for the majority of the studied cases.

Pop. Size	D	SPSO	$\epsilon_0 = 0.2$	$\epsilon_0 = 0.4$	$\epsilon_0 = 0.6$	$\epsilon_0 = 0.8$
20	10	96.1715	**33.6479**	49.0012	37.8162	54.4093
	20	214.6764	115.4576	**113.5109**	176.0168	305.1259
	30	316.4468	**134.5655**	594.9838	1412.8	2750.7
40	10	70.2139	48.2263	24.8386	**17.6052**	41.4613
	20	180.9671	**53.8433**	108.4944	147.0804	208.8547
	30	299.7061	**77.5939**	210.4943	226.3	651.6
80	10	36.2945	27.9244	35.0499	**14.8248**	28.8661
	20	87.2802	**29.7177**	53.5576	85.9102	168.5813
	30	205.5596	**65.2995**	205.6310	159.2	325.2
160	10	24.4477	**14.5572**	19.0323	28.1880	25.8223
	20	72.8190	**44.0544**	56.262	68.3429	92.1108
	30	131.5866	**83.0503**	123.3456	145.9640	134.2

Table 2: Mean fitness values for the Rosenbrock function

6.2 Second experiment

Table 5, Table 6 and Table 7 respectively show the mean fitness values for the Rosenbrock, the Rastrigin and the Griewank function when position is dampened according to Eq. 7.

In this case, a variant of the ϵ-PSO algorithm was always better than SPSO for every combination of test function, population size and dimension size. In the Rosenbrock and Griewank functions, the difference in the mean fitness values is considerable. In the Rastrigin function, however, the difference between the best of the ϵ-PSO variants and the standard algorithm is slight. Finally, leaving out the case of the Rastrigin function (in which $\epsilon_0 = 0.2$ was always better), the optimal ϵ_0 parameter seems to follow complex patterns.

7 CONCLUSIONS AND FUTURE WORK

ϵ-PSO is a profitable algorithm, with similar or better results than the standard algorithm, at least for the chosen set of test functions. The advantage of the proposed algorithm over the original one becomes stronger when position damping is needed due to physical or logical limits to the variables involved. In this case, the performance of ϵ-PSO is notably better than the standard PSO for an appropriate initial value of ϵ. It is believed that part of the success of our algorithm to deal with such problems reside in the fact that, when ϵ still has an important value, the particles are forced to explore the space but are kept effectively confined in the specified search area, without the explosion shown by SPSO when w is large.

Here, we have used a linear decreasing schedule for the exploration parameter ϵ. The results suggest that an initial value of 0.2 could work well in some cases, though the optimal initial value seems to be a non-trivial function of the problem, the number of particles and the dimension size.

Future research must study the causes of the poor performance of standard PSO when position damping is required, as well as investigate real problems for which ϵ-PSO could be significantly better suited than the standard algorithm. A more profound, both theoretical and empirical study of the dynamics of the new algorithm is also required.

Pop. Size	D	SPSO	$\epsilon_0 = 0.2$	$\epsilon_0 = 0.4$	$\epsilon_0 = 0.6$	$\epsilon_0 = 0.8$
20	10	5.5572	**5.0946**	5.8262	8.8894	11.1460
	20	**22.8892**	23.3572	26.8212	35.3031	42.2503
	30	**47.2941**	58.1469	60.6441	77.1062	98.4390
40	10	3.5623	**3.2043**	4.6750	6.4897	8.0282
	20	**16.3504**	16.9558	22.1923	26.5342	32.4187
	30	**38.5250**	42.1334	49.8242	56.6610	80.2835
80	10	2.5379	**2.2884**	3.4024	5.1905	7.0442
	20	**13.4263**	14.1720	16.6663	19.5479	26.7067
	30	**29.3063**	32.7560	39.9838	50.3347	57.8298
160	10	1.4943	**1.1145**	2.4756	3.0857	4.2357
	20	10.3696	**10.1689**	11.9176	17.4540	19.5432
	30	**24.0864**	24.5720	32.3308	38.1781	49.7261

Table 3: Mean fitness values for the Rastrigin function

Pop. Size	D	SPSO	$\epsilon_0 = 0.2$	$\epsilon_0 = 0.4$	$\epsilon_0 = 0.6$	$\epsilon_0 = 0.8$
20	10	0.0919	**0.0835**	0.1024	0.1309	0.1592
	20	0.0303	**0.0193**	0.0245	0.0215	0.0386
	30	0.0182	**0.0116**	0.012	0.0214	0.2083
40	10	0.0862	**0.0622**	0.0861	0.1348	0.1558
	20	0.0286	**0.0175**	0.0282	0.0308	0.0276
	30	0.0127	**0.0119**	0.0122	0.0133	0.0215
80	10	0.0760	**0.0657**	0.0904	0.1107	0.1106
	20	0.0288	0.0272	0.0270	0.0277	**0.0254**
	30	0.0128	0.0115	**0.0096**	0.0136	0.0122
160	10	0.0628	**0.0522**	0.0653	0.0927	0.0926
	20	0.0300	**0.0182**	0.0247	0.0289	0.0239
	30	0.0127	0.0112	0.0125	**0.0085**	0.0122

Table 4: Mean fitness values for the Griewank function

Pop. Size	D	SPSO	$\epsilon_0 = 0.2$	$\epsilon_0 = 0.4$	$\epsilon_0 = 0.6$	$\epsilon_0 = 0.8$
20	10	40526	**38.0016**	445.2396	250	73.9138
	20	21395	20272	**537**	1173	1048.7
	30	61576	**146**	197	61777	3309.6
40	10	514	422.7047	38.9437	**29.4879**	274.296
	20	21163	**510**	20533	901.2547	20735
	30	41643	**336**	360	780.2135	21066
80	10	264	**21.0497**	31.0101	24.2766	30.2339
	20	41338	20678	20543	20558	**20539**
	30	60988	20419	**787**	21124	21391
160	10	241	23.431	25.152	37.8196	**15.1126**
	20	922	1258	867	**287.2525**	1107
	30	1375	**551**	20550	765.4065	21012

Table 5: Mean fitness values for the Rosenbrock function (with position damping)

Pop. Size	D	SPSO	$\epsilon_0 = 0.2$	$\epsilon_0 = 0.4$	$\epsilon_0 = 0.6$	$\epsilon_0 = 0.8$
20	10	5.3559	**5.1155**	6.3868	7.9076	10.6099
	20	24.5782	**22.5863**	26.5219	29.9845	37.1679
	30	62.8768	**53.0729**	53.6623	67.2835	84.6361
40	10	4.2992	**3.3028**	4.6335	6.3629	7.3703
	20	17.6285	**17.0139**	17.0616	27.5237	31.8036
	30	47.5263	**42.2028**	46.0661	53.6205	61.2417
80	10	2.4083	**2.2885**	3.2769	4.0815	6.0558
	20	14.4308	**12.2411**	14.2891	21.1323	24.7033
	30	36.9257	**31.5418**	36.3210	43.8105	49.5687
160	10	1.4741	**1.2139**	2.3101	3.5597	4.2966
	20	10.7648	**9.4924**	11.6179	15.7373	19.4141
	30	28.0666	**24.1691**	28.0846	36.9575	41.5027

Table 6: Mean fitness values for the Rastrigin function (with position damping)

Pop. Size	D	SPSO	$\epsilon_0 = 0.2$	$\epsilon_0 = 0.4$	$\epsilon_0 = 0.6$	$\epsilon_0 = 0.8$
20	10	7.3164	5.5028	3.7315	1.9634	**0.1852**
	20	104.7478	70.4624	48.8585	16.2818	**7.2608**
	30	245.4516	169.6974	142.6833	66.8350	**18.1517**
40	10	1.8959	1.8680	**0.0837**	0.1167	0.1545
	20	50.6074	39.8036	14.4900	**0.0338**	0.0390
	30	158.8310	142.6435	74.1010	21.6965	**3.6290**
80	10	0.0806	**0.0641**	0.0658	0.0984	0.1264
	20	25.3326	16.2763	3.6344	**0.0223**	0.0250
	30	86.6811	95.7091	39.7482	14.4802	**0.0115**
160	10	0.0677	**0.0549**	0.0652	0.0798	0.0993
	20	7.2247	1.8288	**0.0258**	0.0275	0.0321
	30	55.9765	57.7800	25.3011	3.6337	**0.0116**

Table 7: Mean fitness values for the Griewank function (with position damping)

References

[1] Clerc, M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Proc. Congress on Evolutionary Computation*, (1999) Washington, DC, pp 1951-1957.

[2] Clerc, M. and Kennedy, J. The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space. *IEEE Transactions on Evolutionary Computation*, (2002), vol. 6, p. 58-73.

[3] Kennedy, J. and Eberhart, R. C. Particle swarm optimization. *Proc. IEEE International Conference on Neural Networks, IV*, (1995), pp. 1942-1948.

[4] Kennedy, J. and Eberhart, R. C. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, CA, (2001).

[5] Lvbjerg, M., Rasmussen, T.K. and Krink, T. Hybrid ParticleSwarm Optimizer with Breeding and Subpopulations. *In Pro-ceedings of the Genetic and Evolutionary Computation Con-ference (GECCO)*, (2001). San Francisco.

[6] Peram, T., Veeramachaneni, K., and Mohan, C. K. Fitness-distance-ratio based particle swarm optimization. *Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2003)*, (2003) Indianapolis, Indiana, USA. pp. 174-181.

[7] Shi, Y. and Eberhart, R. C. Parameter selection in particle swarm optimization. *Proceedings of the 1998 Annual Conference on Evolutionary Programming*, (1998), San Diego, CA.

[8] Shi, Y. and Eberhart, R. C. A modified particle swarm optimizer. *Proceedings of the IEEE International Conference on Evolutionary Computation*, (1998), pp. 69-73. Piscataway, NJ: IEEE Press.

[9] Shi, Y. and Eberhart, R. C. Empirical study of particle swarm optimization. *Proceedings of the 1999 Congress on Evolutionary Computation*, (1999), pp. 19451950. Piscataway, NJ: IEEE Service Center.

[10] Shi, Y. and Eberhart, R. C. Fuzzy adaptive particle swarm optimization, *Proc. Congress on Evolutionary Computation*, (2001), Seoul, Korea. Piscataway, NJ: IEEE Service Center.

[11] Shi, Y. and Eberhart, R. C. Particle swarm optimization with fuzzy adaptive inertia weight. *Proceedings of the Workshop on Particle Swarm Optimization*, (2001), Indianapolis.

[12] Sutton, R. and Barto, A. *Reinforcement Learning: An Introduction*, (1998). MIT Press/Bradford Books.

[13] Watkins, C. J. C. H. *Learning from Delayed Rewards*, PhD thesis, Cambridge University, Cambridge, England (1989).

[14] Xie, X., Zhang, W. and Yang, Z. Hybrid Particle Swarm Optimizer with mass Extinction. *International Conference on Communication, Circuits and Systems (ICCCAS)*, (2002). Chengdu, China.

[15] Xie, X., Zhang, W. and Yang, Z. A Dissipative Particle Swarm Optimization. *IEEE Congress on Evolutionary Computation*, (2002). Honolulu, Hawaii, USA.

[16] Xie X F, Zhang W J, Yang Z L. Adaptive particle swarm optimization on individual level. *Int. Conf. on Signal Processing*, (2002).

A Alternativa Construtivista em Inteligência Artificial

Filipo Studzinski Perotto
Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
fsperotto@inf.ufrgs.br

Luís Otávio Alvares
Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
alvares@inf.ufrgs.br

Abstract

This paper intends to start ideas about an agent learning architecture based on Constructivist AI approach. First we show how Artificial Intelligence can incorporate some concepts from Jean Piaget's psychology to form Constructivist AI as a new AI paradigm. Then we present details about the proposed architecture, arguing about provided agent autonomy. Finnaly we shortly relate experimental results.

Keywords: cognitive agent, constructivist artificial intelligence.

Resumo

A pesquisa apresentada neste artigo tem por objetivo apontar passos iniciais para a definição de uma arquitetura de aprendizagem para um agente computacional inteligente fundamentada na abordagem da IA Construtivista. Inicialmente discutiremos como a incorporação de alguns conceitos da Teoria Psicológica de Jean Piaget possibilitam o estabelecimento de uma IA Construtivista como um novo paradigma. A seguir, apresentamos detalhes sobre a arquitetura proposta, argumentando sobre a autonomia do agente. Finalmente, relatamos brevemente alguns experimentos.

Palabras claves: agente cognitivo, inteligência artificial construtivista.

1. O DESENVOLVIMENTO DA IA COMO DISCIPLINA CIENTÍFICA

O marco fundador da Inteligência Artificial (IA) foi a conferência de Dartmouth em 1956. Porém o intento de reproduzir essa característica distinta do ser humano – a inteligência – data de muito antes, havendo registros históricos desse pensamento desde a antiguidade na literatura, na engenharia e na filosofia. Havia mesmo, na metade do século XX, um contexto propício para o florescimento da Inteligência Artificial como ciência. De um lado, a inquietante curiosidade do ser humano por conhecer a si mesmo serviu como motivação para irmos em busca deste empreendimento pretensioso: desvendar os mistérios relacionados à inteligência e construir uma inteligência artificial. De outro lado, as Ciências Humanas e as Ciências da Computação forneciam um ferramental interessante e um quadro otimista para esse surgimento. [McCorduck 1979], [Perotto 2002].

O movimento desta ciência, nesses seus 50 anos de existência, poderia ser resumido da seguinte maneira: uma primeira década marcada por muito otimismo, quando os pesquisadores encarregaram-se de vislumbrar as possibilidades, estabelecendo as grandes metas e sonhos da IA através de promessas ambiciosas. Nos anos que se seguiram, as dificuldades, os fracassos e a falta de resultados convincentes marcaram uma desilusão. Apesar de todo o empenho dos pesquisadores, a constatação foi contundente: reproduzir no computador os comportamentos inteligentes do ser humano seria uma tarefa muitíssimo mais árdua do que se imaginava. A necessidade de concretizar suas proposições fez com que a IA passasse por um processo de fechamento e especialização: os cientistas voltaram seus esforços para o aperfeiçoamento das técnicas que já haviam sido propostas e as pesquisas tornaram-se mais específicas e menos

ambiciosas. Esse fechamento teve seu ápice na década de 1970, quando o pragmatismo tomou o lugar dos sonhos da década de 1950 e 1960, e levou os pesquisadores a uma busca por resultados concretos. Se esse pragmatismo teve seu lado positivo, pois finalmente os conhecimentos adquiridos pela disciplina passavam a ser utilizados em larga escala na prática, também teve seu lado negativo, pois desarticulou a IA enquanto um grande projeto de compreensão da inteligência. Os paradigmas clássicos (IA Simbólica e IA Conexionista), à medida que iam sendo exaustivamente explorados, começavam a dar sinais de saturação. O desenvolvimento da IA em cima das mesmas e velhas técnicas gradativamente mostrava seus limites. Dessa maneira, um movimento de reabertura tornou-se necessário, em busca de inspiração para a construção de modelos alternativos. As décadas de 1980 e 1990 viram surgir, então, propostas de novos paradigmas, de novas técnicas (muitas delas nascidas de hibridização das técnicas clássicas), e de uma busca por novas concepções teóricas, provenientes de um renovado contato com outras disciplinas. [Teixeira 2000], [Coelho 1999], [Barreto 1999].

2. A CONCEPÇÃO CONSTRUTIVISTA DA INTELIGÊNCIA ARTIFICIAL

Um dos frutos do processo de renovação da IA é o surgimento da *Inteligência Artificial Construtivista* como um novo paradigma que, em linhas gerais, engloba todos os trabalhos desta ciência que fazem referência à Teoria Psicológica Construtivista, fundamentada na obra de Jean Piaget. A concepção construtivista da inteligência artificial foi definitivamente estabelecida por Gary Dresher [1991], embora trabalhos anteriores já ensaiassem essa aproximação. A IA Construtivista permanece ainda hoje como um paradigma alternativo, não tendo conquistado lugar de destaque entre os pesquisadores da área, que em sua maioria continuam ligados aos paradigmas clássicos. Porém, mesmo sendo uma proposta pouco explorada, ainda em busca de legitimidade e consolidação, a IA Construtivista já se mostrou apta a fornecer contribuições significativas para a disciplina, constituindo-se no mínimo como um caminho alternativo que merece ser explorado. Discussões mais aprofundadas sobre questões teóricas do encontro entre a Psicologia Construtivista e a Inteligência Artificial podem ser encontradas em [Costa 1986], [Drescher 1991], [Boden 1983] e [Inhelder & Cellèrier 1996].

A mudança paradigmática requer três movimentos. Primeiro uma desantropomorfização da IA, pois a noção de inteligência não deve estar atrelada à inteligência humana. A inteligência humana é uma instância dentre uma gama de sistemas inteligentes possíveis. O ser humano é dotado de uma constituição sistêmica própria (ou seja, um conjunto particular de objetivos, funções orgânicas, tipos de interação com o meio, etc), e por isso possui um tipo de inteligência específico. O computador habita outro tipo de ambiente, e constitui-se num sistema distinto do humano, por isso seu processo de adaptação resultará num tipo de inteligência própria. Ao perceber que a máquina faz um uso particular de suas estruturas cognitivas, atribuindo a elas significados que só têm sentido na sua vivência, podemos ter uma visão da IA como o estudo da inteligência de máquina enquanto um fenômeno natural na máquina, guiada por suas próprias experiências e necessidades funcionais, e não como a reprodução do comportamento humano. [Rocha Costa 2003]

O segundo movimento é a adoção integral do conceito de agente situado, incorporado e autônomo, em substituição à idéia clássica de sistema computacional. O sistema computacional da teoria clássica estabelece uma relação linear do tipo *entrada–processamento–saída*. Pensar através da metáfora do agente leva-nos a considerar o ambiente em que o agente está situado, e um fluxo de interação contínua entre um e outro. O agente percebe seu ambiente através de sensores específicos, que lhe fornecem um acesso parcial e elementar ao mundo. Essa percepção tem de ser interpretada pelo agente e, como é parcial, precisa ser completada pelo conhecimento do agente. O agente também possui capacidade de interferir nesse ambiente através de atuadores específicos. O ambiente pode ser complexo, dinâmico, não-estacionário e pode esconder suas regularidades de tal forma que somente estruturas de alto-nível possam capturá-las. O agente é autônomo, tanto no que diz respeito às tomadas de decisão, quanto à própria construção cognitiva. Não é o ambiente que guia diretamente as ações do agente e seu processo de aprendizagem, mas sim a tensão entre ambos. São dois pólos igualmente considerados: o ambiente enquanto palco das experiências, e o agente, enquanto sujeito que organiza, compreende e dá sentido a essas experiências. O agente torna-se incorporado. Os sensores e atuadores fazem parte de um corpo, que realiza a interface entre a mente e o exterior. O corpo constitui-se também num ambiente interno, com ciclos, metabolismos e propriedades particulares. A mente de um agente, da mesma forma deve incluir além de um sistema cognitivo, um sistema emocional que dê sentido valorativo às suas vivências. Tais elementos são elementos fundamentais para garantir uma verdadeira autonomia para o agente, pois assim toda a forma de intencionalidade é um resultado que emerge do funcionamento de seus próprios mecanismos. Os objetivos do agente tornam-se conseqüência deste funcionamento [Rocha Costa e Dimuro 2003], [Sloman 1999], [Pfeier e Sheir, 1999], [Franklin, 1997], [Ledoux, 1996], [Damásio 1994].

Finalmente, o terceiro movimento é o abandono da concepção tradicional de aprendizagem como sendo uma conformação do agente às imposições do meio. A psicologia construtivista é uma teoria sobre desenvolvimento

cognitivo, e por isso mesmo a inteligência artificial construtivista transforma a aprendizagem de máquina em desenvolvimento de máquina. A diferença é justamente a valorização do papel do agente, enquanto organizador da experiência.

O construtivismo sugere que o sujeito, ao nascer, possui relativamente poucas estruturas cognitivas, porém já incorpora mecanismos de aprendizagem que lhe permitem construir novos conhecimentos. Esse conhecimento surge como fruto de uma interação ativa com o mundo exterior [Montangero 1998]. A inteligência pode ser analisada através da capacidade de adaptação do sujeito ao seu meio, pois para adaptar-se ao mundo, o sujeito precisa descobrir quais são suas regularidades, e quais são os efeitos que pode esperar de suas ações. Conseqüentemente, apenas uma inteligência complexa (capaz de construções cognitivas aprimoradas) será capaz de dar conta de um mundo complexo. Assim, a adaptação, por si, é apenas o aspecto exterior, observável, de um processo interior, auto-regulado, de organização e ampliação do conhecimento [Piaget 1975a].

Piaget chama de 'esquema' a estrutura de conhecimento elementar. Um esquema é aquilo que, numa ação, pode ser diferenciado, generalizado e aplicado a outras situações [Montangero 1998]. O conjunto de esquemas do sujeito representa, a cada momento, aquilo que ele acredita e espera do mundo em que vive. Toda a experiência é compreendida e interpretada através dos esquemas. Por isso mesmo, o primeiro gesto do sujeito, diante de uma nova situação, é o de tentar submetê-la ao conjunto de esquemas que possui, ou seja, compreender o novo a partir do que já é sabido. A realidade, no entanto, pode por vezes não se mostrar adequada a isso. Os esquemas podem falhar, e assim obrigar o sujeito a modificá-los para corrigi-los [Piaget 1975b]. É um processo de regulação das interações entre o sujeito e seu ambiente, onde ocorre 'assimilação' quando os esquemas do sujeito são capazes de tratar uma situação nova que se apresenta, e 'acomodação' quando os esquemas do sujeito não são capazes de dar uma resposta satisfatória à nova situação e assim a pressão do meio leva o sujeito a modificar-se, diferenciando progressivamente seus esquemas a fim de dar conta da sua realidade e tornar-se mais adaptado a ela [Piaget 1978].

Ao mesmo tempo em que o processo de construção cognitiva deve ser capaz de corrigir-se para integrar as novidades que se apresentam, também deve ser capaz de modificar-se preservando o conhecimento anteriormente adquirido. Cada acomodação amplia a capacidade de assimilação do sujeito. Portanto, a atividade cognitiva tende, num ambiente estável, a alcançar um ponto de equilíbrio [Piaget 1975a]. Quanto mais diminui a defasagem entre o novo e o conhecido, menor é o impacto das novidades no equilíbrio do sistema, e mais fácil é a adaptação [Piaget 1975a].

No início de sua vida, o sujeito percebe apenas "quadros sensoriais" (percepções instantâneas do campo visual, auditivo, tátil, cinestésico, gustativo...). Esses quadros não têm nenhuma relação lógica implícita, sendo, a priori, desconexos. A partir desse universo de "quadros sensoriais" a inteligência irá construir noções elementares, estabelecer relações, encontrar regularidades, e eventualmente estabelecer em um mundo objetivo, substancial, espacial, temporal, regular, exterior e independente de si. O "real" vai ser construído nesse movimento de adaptação pela crescente coerência entre os esquemas. [Piaget 1975b].

Uma coisa é aprender um elemento novo, um resultado de uma ação específica, que embora sendo novo não requer nenhuma maneira nova de compreensão. É uma instância nova de aplicação de um esquema já conhecido, e ainda que essa aprendizagem possa engendrar processos de acomodação, a transformação permanece ligada à utilização de estruturas já construídas para tratar situações análogas. Outra coisa é construir uma ferramenta intelectual nova, subir mais um andar na construção de formas de compreensão das situações, e assim, ganhar poder de assimilação não por recorrer à similaridade com o que já se sabe, mas, ao contrário, por utilizar a criatividade para conceber uma nova estrutura, uma nova maneira de pensar e interpretar as situações [Piaget, 1978].

Existe mesmo um processo de complexificação das estruturas cognitivas, que vai na direção de poder representar conhecimentos cada vez mais abstratos e conceituais, em oposição àqueles mais sensório-motores e imediatos. A capacidade dedutiva vai sendo ampliada à medida que o sujeito constrói sistemas mais abstratos de compreensão que, por serem mais flexíveis, tornam-se mais equilibrados e estáveis. Se o sujeito consegue compreender o funcionamento de um dado domínio, isto é, se ele possui um sistema de conceitos e operações que é capaz de representá-lo e representar seu funcionamento, em oposição a simplesmente conhecer uma série linear de resultados, então o sujeito é capaz de antecipar um resultado sem nunca tê-lo visto, apenas por dedução.

3. ESTABELECENDO AS BASES PARA UM AGENTE CONSTRUTIVISTA

A primeira proposta de arquitetura de agente inspirada no Construtivismo está descrita em [Drescher, 1991]. É neste trabalho que se estipula a versão computacional do 'esquema' piagetiano, que é implementado como um registro composto por um contexto, uma ação e uma expectativa (figura 1). A aprendizagem ocorre pela criação de novos esquemas, uma operação que se realiza através de análise estatística da correlação entre elementos do contexto

observado antes da aplicação de uma ação e a alteração desses elementos como resultado desta ação. O modelo de Drescher inspirou outros trabalhos construtivistas em IA, como o de Wazlawick [1993], que substitui o cálculo de correlações por um sistema de esquemas inspirado em algoritmos genéticos [Holland 1992] e no modelo de redes auto-organizáveis [Kohonen 1989]. Muñoz [1999] insere um sistema de planejamento no agente de Wazlawick. Balpaeme, Steels e Looveren [1998], usam a mesma estratégia para conseguir a emergência de categorias e conceitos numa comunidade de agentes. Birk e Paul [2000] inserem programação genética no Agente de Drescher. Outros modelos também são propostos em [deJong 1999], [Müller e Rodriguez 1996], mas nenhum foge drasticamente da idéia central da arquitetura proposta por Drescher.

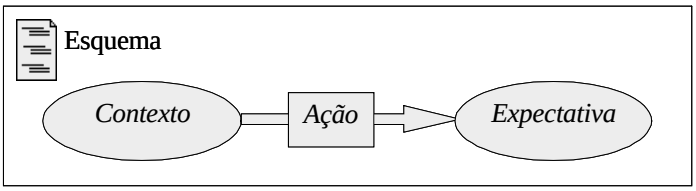


Figura 1. Um Esquema.

Os modelos citados até aqui, e que compõe praticamente todo o quadro de propostas da IA Construtivista, utilizam métodos estatísticos para análise de freqüências como parâmetro de aprendizagem (seja através do cálculo de correlação, dos algoritmos genéticos, dos métodos de reforço ou do ajuste de pesos). Por um lado, a abordagem estatística possui vantagens, pois garante a solução para os casos comuns, e evita que ruídos prejudiquem a constatação de determinadas regularidades. Quando somada a métodos evolutivos (como algoritmos genéticos), permite ainda uma adaptação relativamente rápida nos casos em que o agente muda de um sub-ambiente para outro. Por outro lado, a desvantagem é que justamente um evento raro pode representar o contra-exemplo necessário para romper uma certeza e ampliar o conhecimento, e a estatística tende a desprezar tal evento [Langley e Zytkow, 1990]. Além disso, no caso dos algoritmos genéticos, pode acontecer de o agente nunca estabilizar seu conhecimento, se houver uma intermitente mudança entre um sub-ambiente e outro, caso em que o agente ficará construindo, destruindo e reconstruindo indefinidamente seu conhecimento [Booker, Goldberg, Holland, 1990].

O modelo que estamos propondo difere dos anteriores porque é baseado num mecanismo indutivo de assimilação e acomodação. Uma primeira versão já foi apresentada em [Perotto e Álvares, 2004]. Cada experiência particular pode ser a desencadeadora de toda uma transformação cognitiva do agente, se seus resultados forem muito diferentes do esperado, mesmo que isso seja um evento raro. Essa opção garante ao agente a possibilidade de construir um conhecimento exaustivo sobre as regularidades do mundo, em contrapartida, exige a presença de um mecanismo robusto para detectar ruídos e impedir que eles destruam um conhecimento correto.

A arquitetura do agente pode ser dividida em *Corpo* e *Mente*. O corpo possui uma lista de sensores (entradas) e atuadores (saídas), que fazem a interface com o mundo exterior. O corpo também possui propriedades corporais e atividades metabólicas que lhe dão a característica de um organismo dinâmico. A mente pode ser dividida em *Sistema Cognitivo* e *Sistema Emocional*. O sistema emocional interage com o sistema cognitivo e é o responsável por orientar as ações do agente, conferindo um significado afetivo ao conhecimento, e habilitando o agente a ter seus próprios objetivos. As emoções são estados internos do agente que atribuem valor a sua própria condição atual, que é identificada cognitivamente. A mente também possui uma interação com o mundo interno do corpo, através de sensores e atuadores internos que percebem e modificam sua atividade orgânica (figura 2).

O sistema cognitivo armazena um conjunto de esquemas, e possui um mecanismo de atividade que é responsável tanto pelo processo de tomada de decisão (a ação a ser realizada a cada instante) quanto pelo processo de aprendizagem (aprimoramento do conjunto de esquemas). O esquema é a unidade elementar de conhecimento no sistema, e é composto pela trinca {Contexto, Ações, Expectativas} (como já visto na figura 1). O vetor de *Contexto* representa a situação em que o agente se encontra em cada momento (dada pelos sensores). O vetor de *Ação* representa a ação que será executada pelo agente (através de seus efetores) no caso do esquema ser ativado. O vetor de *Expectativas* reflete a situação que o esquema espera encontrar depois da aplicação de suas ações. Um esquema tem uma boa capacidade de previsão se costuma ter suas expectativas correspondidas após a execução de suas ações.

O processo de tomada de decisão é feito pela função de assimilação, onde são indicados, entre os esquemas do conjunto de esquemas, aqueles que estão aptos a serem aplicados à situação atual do agente (identificada pela percepção), e então entre esses esquemas indicados, é escolhido aquele que será realmente aplicado (que vai realizar sua ação). Durante a atividade do agente, a cada instante, o sistema cognitivo verifica o contexto de todos os esquemas, excitando

aqueles que forem compatíveis com a situação percebida pelos sensores. O mecanismo consulta, então, o sistema emocional, e escolhe entre os esquemas excitados aquele que será ativado.

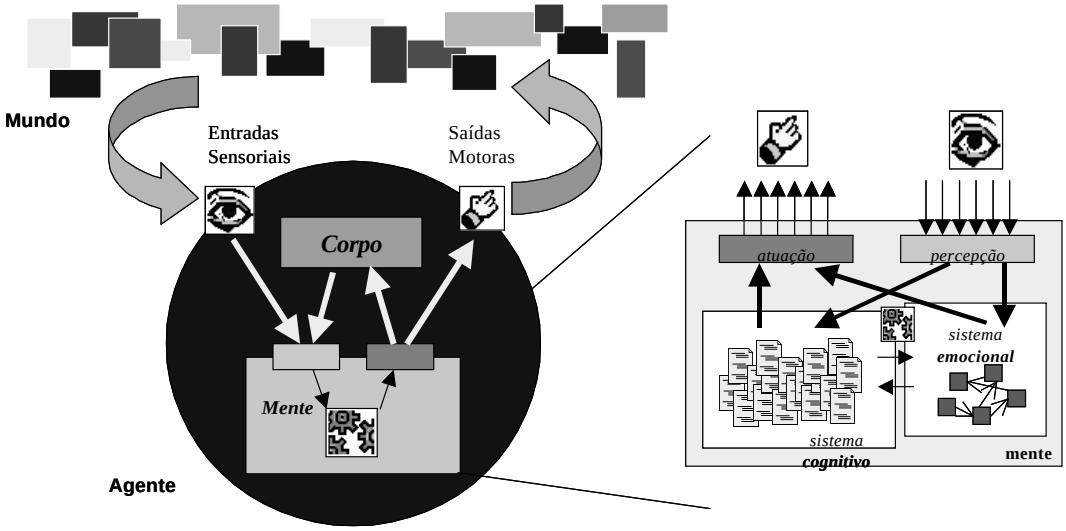


Figura 2. Arquitetura do Agente. À esquerda, o agente como um organismo composto por mente e corpo interagindo com o ambiente. À direita, detalhe da mente, composta por sistema cognitivo e sistema emocional.

A trinca {Contexto, Ação, Expectativa} é composta por vetores de elementos que podem assumir três valores: verdadeiro, falso ou "não importa" (representado, respectivamente, por '1', '0' e '#'). O valor verdadeiro indica a presença do elemento, o valor falso indica sua ausência. O valor "não importa" serve para generalizar um elemento no esquema, indicando que a compatibilidade do esquema não depende do seu valor. Por exemplo, um esquema com vetor de contexto = (0,1,#) é capaz de assimilar as situações (0,1,0) e (0,1,1). Há compatibilidade entre um esquema e uma determinada situação quando o vetor de contexto do esquema tem todos os seus valores verdadeiros e falsos equivalentes no vetor de percepção do agente. Note-se que a compatibilidade não compara valores "não importa". Essa estratégia foi proposta por Holland [Holland] e possibilita uma flexibilização na representação de conjuntos de situações. Assim não há necessidade de considerar cada estado possível das combinações de elementos da percepção, justamente o principal problema de complexidade que aflige a maioria dos algoritmos de aprendizagem por reforço.

O conhecimento do agente vai sendo construindo pelo desenvolvimento do conjunto de esquemas. Por um lado, a assimilação é responsável por criar novos esquemas para situações novas. Por outro, a acomodação ajusta e corrige esses esquemas, seja por diferenciar domínios de atuação (dividindo um esquema em mais esquemas), seja por generalizar suas expectativas. O esquema é corrigido após cada ativação, e como essa correção atua no sentido de aumentar a capacidade de previsão do esquema.

Todo o problema de aprendizagem incremental que combina a aprendizagem e o desempenho no mesmo momento precisa enfrentar o dilema da exploração. Se o agente muito cedo preferir utilizar os esquemas que já possui, sem arriscar-se a experimentar ações cujo resultado é incerto, correrá o risco de ficar preso num máximo local, que pode mesmo ser uma solução ruim. Por outro lado, só explorar o ambiente sem utilizar o conhecimento adquirido para melhorar seu desempenho não é algo inteligente. A saída é colocar um índice de curiosidade que leva o agente a explorar eventualmente ações alternativas, quando há ações alternativas a serem exploradas.

4. ALGORITMOS DE APRENDIZAGEM

Classicamente em Aprendizagem de Máquina existem duas abordagens para relacionar conjuntos de situações: bottom-up e top-down, ou seja, começar por uma representação completamente específica e ir gradativamente generalizando-a e integrando-a a outras representações que se mostrem comuns, ou o inverso, começar por uma representação completamente geral, e ir especializando-a e diferenciando-a conforme necessário. No nosso caso será necessário combinar ambas as formas, uma vez que o aprendizado de um agente é algo um pouco diferente de uma classificação.

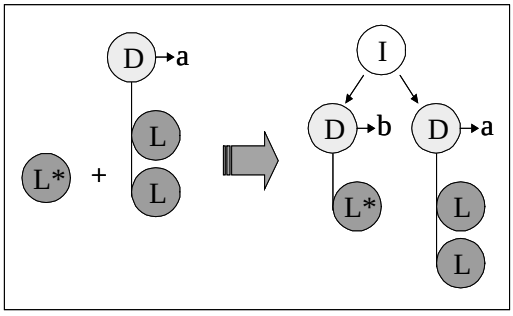
O caminho que o conhecimento do agente percorre é o seguinte: primeiro, quando ocorrem desequilíbrios, o agente aumenta o número de esquemas, diferenciando-os por especializar gradativamente seus vetores de contexto. Quando isso não é possível, então o sistema cognitivo se obriga a reduzir suas expectativas, generalizando-as gradativamente. Por fim o conjunto de esquemas, após esse processo de diferenciação e generalização das expectativas pode ainda permitir integrações de esquemas que acabaram tornando-se equivalentes, o que reduz o número de esquemas do conjunto. Assim, o processo de aprendizagem do agente construtivista proposto acontece através da função de acomodação, que possui 3 métodos para o refinamento do conjunto de esquemas: *Diferenciação*, *Ajuste de Expectativas* e *Integração*.

A mente do agente é iniciada com um único esquema que possui um contexto completamente geral, e uma expectativa indefinida. Ter o contexto completamente geral significa que qualquer situação será compatível com ele, e que, portanto, este esquema inicial está em princípio capacitado para assimilar todas as experiências. Na primeira ativação do esquema a expectativa se define como o próprio resultado encontrado após a execução da ação. Assim todo o esquema recém-definido terá a expectativa completamente específica, inversamente ao que ocorre com o contexto.

Muito provavelmente esse grande esquema geral rapidamente encontrará uma nova situação onde o resultado da mesma ação será diferente de sua expectativa. Quando um esquema não tem suas expectativas correspondidas após uma ativação, a primeira estratégia é tentar dividir o seu domínio de atuação entre esquemas mais específicos. O método de diferenciação pega o contexto de um esquema que tenha falhado, e cria, a partir dele, vários esquemas pela especialização do contexto (substituição de algum dos valores '#', pelos valores 1 e 0). Assim, nestes novos esquemas estará distribuído tanto o conhecimento sobre as situações para as quais o esquema anterior funcionava bem, quanto novas soluções para a situação falha (figura 3).

Imaginemos um exemplo ilustrativo. Suponha um agente que habita um mundo cheio de formas geométricas coloridas, e que as ações que pode executar no mundo são deslocar-se em busca de outros objetos e tocar nos objetos que encontra. Ao tocar num cubo vermelho, o agente observa a transformação deste em uma pirâmide azul. Se esta á a primeira experiência do agente nesse mundo, ele preservará apenas um esquema que supõe o seguinte: "para qualquer forma geométrica de qualquer cor (contexto geral), a ação de tocar levará à transformação dessa forma geométrica em uma pirâmide azul (expectativa específica)". A seguir, diante de outro objeto, suponhamos um cilindro amarelo, o agente poderá aplicar o mesmo esquema, afinal, trata-se de um objeto que faz parte da classe definida no vetor de contexto do esquema por "qualquer forma geométrica de qualquer cor". Porém ao tocá-la, surge uma esfera verde. A expectativa era do surgimento de uma pirâmide azul. O mecanismo utiliza o método de diferenciação para preservar o esquema no escopo onde funcionava bem, e criando outro esquema para a nova situação. O algoritmo precisa escolher um elemento diferenciador, no caso, os dois elementos (forma e cor) diferenciam a situação, então é escolhido qualquer um deles. Escolhendo-se forma, por exemplo, então agora haveria dois esquemas decisores, um dizendo que "cubos, ao serem tocados, transformam-se em pirâmides azuis", e outro dizendo que "cilindros, ao serem tocados, transformam-se em esferas verdes".

A diferenciação não precisa utilizar apenas elementos da percepção para diferenciar entre as situações. Na verdade, antes disso, é melhor tentar diferenciar pela ação do agente. No exemplo ele fazia apenas um tipo de ação com os objetos (tocar), mas se existissem outras opções (tocar, bater...) seria melhor supor que o resultado diferente seria conseqüência de ter realizado uma ação diferente. Note-se que a diferenciação vai construindo uma árvore, onde cada nível significa a introdução de uma restrição no vetor de contexto para descrever a situação, partindo de um nó raiz geral (que é compatível com todas as situações) chegando até os nós especializados para as classes de situações, que chamamos de esquemas decisores.



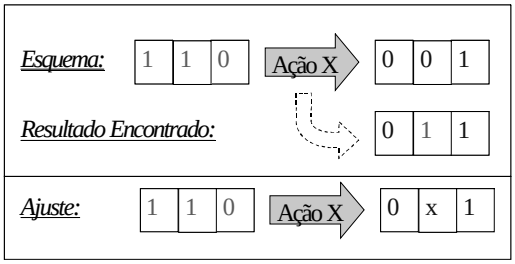

```
Método DIFERENCIAÇÃO

D := Agent.ActivatedScheme;
if not Success(D) then
  E := FindDiffer(D);
  NewD := D.CreateCopy;
  NewD.Context[E] := 0
  D.Context[E] := 1
  NewD.Expect := ActualPerception;
```

Figura 3. Diferenciação. À esquerda, o esquema 'D', com seus casos bem assimilados 'L', onde o resultado da aplicação da ação é 'a'. À direita, o resultado da diferenciação que o transformou em dois esquemas para adaptar-se ao novo caso 'L*'

Quando não é possível diferenciar, então o sistema é obrigado a reduzir as expectativas do esquema. O método de ajuste de expectativas modifica um esquema já existente, a fim de corrigir suas expectativas quando elas falham. O método simplesmente compara a expectativa do esquema ativado no último turno, e verifica se o resultado encontrado no ambiente (percebido pelo agente através de seus sensores) foi condizente. Todos os elementos do vetor que estiverem diferentes são trocados para o valor '#' ("não importa"), significando que aquele elemento não pode ser previsto como resultado da ação do esquema, uma vez que seus resultados não são regulares após a aplicação da ação no mesmo contexto. Como o mecanismo sempre cria esquemas com expectativas completamente definidas (iguais ao resultado da primeira aplicação), os esquemas sempre caminham na direção de uma diminuição das expectativas, até encontrar somente as regularidades previsíveis.

Se quisermos continuar ilustrando com o mesmo exemplo, seria o caso de supor o seguinte esquema: "tetraedros laranjas, ao serem tocados, transformam-se em cubos pretos". A princípio este esquema funcionava bem, ou seja, sempre tinha suas expectativas correspondidas. Suponhamos que, numa situação, o agente toca num tetraedro laranja e ele vira um cubo rosa. Não há como diferenciar o contexto pela percepção, pois são perceptivamente iguais. A alternativa nesse caso é reduzir a expectativa, fazendo-se esperar agora apenas um "cubo" (sem definir a cor), e não mais um "cubo preto".



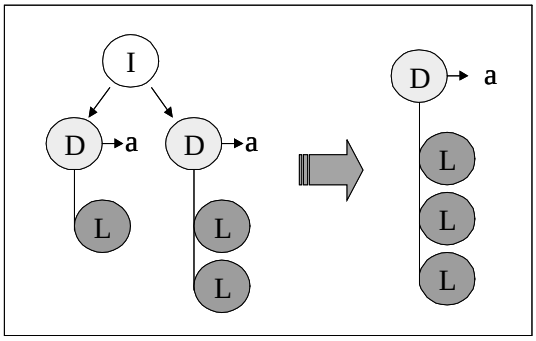

```
Método AJUSTE DE EXPECTATIVAS

S := Agent.ActivatedScheme;
For i := 0 to Percept.Size-1 do
  if S.Expect[i] <> Percept[i] then
    S.Expect[i] := #; {don't care}
```

Figura 4. Ajuste de expectativas de um esquema. O quadro mostra um esquema que teve sua expectativa contestada por um resultado, e em seguida o ajuste.

Finalmente, quando o mecanismo reconhece dois esquemas similares que, porém, tratam situações diferentes, então estes esquemas são unidos pelo método de Integração. Por causa dos ajustes de expectativas, dois esquemas que antes haviam sido diferenciados agora podem ser novamente integrados. Estes dois esquemas possuem a mesma ação e o mesmo conjunto de expectativas, e se diferenciam por algum elemento do vetor de contexto. O método então encontra um contexto mais geral que é compatível com os contextos desse dois esquemas, e a partir dele, cria um novo esquema para substituí-los (figura 5).

No exemplo, seria supor que em determinado momento da execução, existe um nível da árvore que diferencia os cubos pelas suas cores (cubo verde, cubo azul, cubo vermelho...), porém todos esses esquemas acabaram ficando com a mesma expectativa, suponhamos, "ao ser tocado, transforma-se numa forma geométrica branca". Se não é mais preciso diferenciar pelas cores, então elimina esses esquemas mais específicos, e considera a expectativa do nó imediatamente superior da árvore, que seria "um cubo, ao ser tocado, transforma-se numa forma geométrica branca".




```
Método INTEGRAÇÃO

if (D1.Parent = D2.Parent) and
   (D1.Expect = D2.Expect) and
   (D1.Action = D2.Action) then
  I.Expect := D1.Expect;
  Delete(D1);
  Delete(D2)
  NewDecisor(I);
```

Figura 5. Integração. No quadro, à esquerda, dois esquemas 'D' anteriormente diferenciados, mas que agora possuem a mesma expectativa 'a'. No quadro, à direita, o resultado da integração.

5. ESTRUTURAS COGNITIVAS PARA SITUAÇÕES COMPLEXAS

O agente apresentado até aqui, utilizando o modelo de aprendizagem básico descrito na seção anterior, já é capaz de descobrir sozinha uma série de regularidades perceptivas do seu ambiente. Porém ainda é muito simples, limitando-se a um aprendizado de reações, num plano unicamente sensório-motor. A vantagem é que essa arquitetura permite a expansão das capacidades do agente, pela articulação dos métodos básicos com novos métodos. Estes novos métodos fazem parte do desenvolvimento atual da pesquisa, que visam incluir no agente capacidades para tratar situações num grau de complexidade maior. Alguns deles já haviam sido apontados por [Drescher, 1991].

A primeira capacidade a ser acrescentada é a *percepção temporal.* O agente deve ser capaz, não só de identificar regularidades a partir de percepções instantâneas de uma situação, mas também perceber contextos distribuídos no tempo, onde uma seqüência de acontecimentos ou ações forma um conjunto significativo. Essa capacidade é um pré-requisito para que seja possível a emergência de planejamento no agente. A estratégia inicial é incluir uma estrutura de encadeamento entre os esquemas. Isto é, conectar dois esquemas quando a expectativa de um é compatível com o contexto do outro (figura 6). Nem sempre esse encadeamento é tão linear, pois os vetores de contexto e expectativa podem ter elementos indefinidos, mas é possível construir um grafo de todos os possíveis encadeamentos. Esta estrutura permite também considerar "ações compostas", que seriam o primeiro passo de afastamento do sensório-motor em direção ao abstrato. Um esquema utilizando uma ação composta interliga da mesma maneira contexto e expectativa, porém a ação faz referência a uma série de ações que são encadeadas em sub-esquemas. Para ilustrar isso imagine na vida real a ação de "apagar a luz". É uma ação que está num nível abstrato, uma vez que as sub-ações que devem se encadear para realizá-la dependem, por exemplo, de onde eu estou e onde está o interruptor.

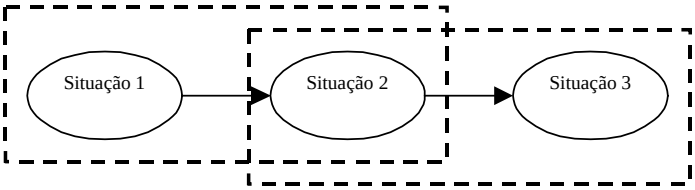


Figura 6. Dois esquemas encadeados, uma vez que a expectativa de um é compatível com o contexto do outro.

Outra capacidade essencial é a *formação de conceitos abstratos.* O agente deve ser capaz de superar suas formas de percepção, não se limitando às meras percepções elementares, sensoriais, mas construindo conceitos mais abstratos. Certos eventos do mundo real só tornam-se inteligíveis a partir da formação de conceitos abstratos. É o caso, por exemplo, da noção de "objeto", que utilizamos para não tratar o mundo perceptivo como um mapa de pontos. A idéia de conceito é amplamente utilizada em IA, porém geralmente está associada com a formação de classes ou agrupamentos. Essas formações são importantes, e o mecanismo de esquemas aqui proposto já permite um tipo de classificação através da própria possibilidade de generalizar um contexto para abranger uma grande gama de situações. Porém é preciso abranger outros tipos de classes, que nem sempre estão relacionadas por propriedades perceptivas, mas sim por propriedades funcionais ou operatórias. Por exemplo, uma caneta e um lápis podem ser perceptivamente bem diferentes, mas são ambos classificados como "objetos que servem para escrever", e nessa

perspectiva, fazem parte de uma mesma categoria de coisas que uma generalização perceptiva nunca alcançaria. Outro exemplo é o caso em que uma folha de papel lisa é amassada, e transformada numa bolinha de papel. Perceptivamente, mais uma vez, os dois objetos são muito diferentes, mas se o agente conhece a operação que transforma a folha de um estado para outro, então pode relacioná-las.

Outra forma de abstração é a criação de propriedades sintéticas, que dizem respeito a estados ocultos das situações. Quando um esquema não funciona numa situação idêntica à outra já experimentada e que antes havia funcionado bem, nem sempre deve obrigar o agente a reduzir as expectativas. Uma possibilidade é que exista alguma propriedade não diretamente observável no ambiente, e que explicaria a diferença de resultados. Assim, é necessário incluir no agente um mecanismo que crie e gerencie essas propriedades sintéticas. Para exemplificar isso, suponha o caso em que você observa duas caixas idênticas, porém ao tentar carrega-las, você percebe que uma é muito pesada e outra é muito leve. Pois bem, o peso não é uma propriedade diretamente observável pela percepção. É muito mais razoável supor a existência de uma propriedade oculta (o peso), do que simplesmente deixar de tentar prever o comportamento da caixa quando levantada. Pode parecer inútil inicialmente manter no sistema uma propriedade sintética que só terá seu valor revelado depois da ação (no caso, você só saberá se a caixa é pesada ou não depois de levantá-la). Mas quando se trata de um sistema interligado de esquemas, então já existem vantagens. Imagine que eu entregue a você duas bolas, uma de futebol e a outra de boliche, e suponha que eu as tenha pintado de forma que, perceptivamente, seja impossível distingui-las. Antes de chutar uma delas, você pode tentar levantá-las para saber quais são os seus pesos.

Desenvolvimento de múltiplos sistemas de conhecimento: o agente deve ser capaz de construir subconjuntos de percepções, que representariam domínios específicos de conhecimento. Trata-se de sistemas de conhecimento dinâmicos, em interação e criadores de significados próprios e sujeitos a adaptações específicas, durante o processo adaptativo do agente.

Se chegamos aqui, estamos a um passo da *construção de sistemas hipotético-dedutivos*. A própria existência das propriedades sintéticas permite o raciocínio através de hipóteses. No exemplo, o agente poderia fazer a seguinte hipótese: “se está aqui for a bola de boliche, então meu pé irá doer se eu chutar”. Com isso, nosso agente desprende-se das formas de aprendizagem indutiva, podendo adquirir conhecimento de forma analítica. Num ambiente complexo, o agente não pode esperar que todos os casos aconteçam para que ele tenha um conhecimento deles. É preciso haver um mecanismo de inferência de conhecimentos a partir do que já se é sabido. Expandindo essa idéia, poderíamos considerar a possibilidade não apenas de fazer hipóteses relativas aos estados ocultos do mundo, mas também hipóteses sobre o próprio funcionamento do mundo. Esse mecanismo deve possibilitar a existência de hipóteses concorrentes, que mais tarde se consolidam, ou não, como novos esquemas. Evidentemente, a possibilidade de que o agente especule hipoteticamente sobre seu universo requer que o sistema cognitivo seja capaz de conviver com contradições temporárias, e incertezas quanto à validade dos próprios esquemas, ao mesmo tempo que deve buscar solucionar tais conflitos cognitivos. Uma saída é a utilização do princípio de competição dos algoritmos genéticos. Sistemas concorrentes que funcionam bem, permanecem, e os que funcionam mal, são eliminados.

O agente também deve ser capaz de separar seus esquemas em domínios distintos, para evitar de misturar problemas separados numa mesma árvore de esquemas. *Construção de operações de transformação entre sistemas* construir operações de transformação regulares entre sistemas e entre esquemas, impedindo assim, que o conhecimento seja reduzido a um banco de registros de casos, e permitindo que o agente não só conheça a regularidade dos fenômenos, mas desenvolva também uma compreensão sistemática das transformações pelas quais o mundo se desenvolve.

Tendo isso em vista, podemos agora pensar em uma alternativa para tratar ruídos. A dificuldade justamente é saber quando um evento raro é algo que deve ser desconsiderado, e quando ele deve desencadear as mudanças no conjunto de esquemas. Como é impossível saber de antemão qual é o caso, o mecanismo precisará guardar hipóteses alternativas. A estratégia é criar ramos especiais no caso de um desses eventos. Um primeiro ramo diferenciado do esquema segue o método normal de ajuste de expectativas. Um segundo ramo cria um esquema probabilístico, que preserva a expectativa sem alterações, mas passa a supor que aquele resultado acontece numa parte das vezes proporcional à probabilidade. O terceiro ramo, então, é o que cria uma propriedade sintética, e supõe que, naquele caso o resultado não deu certo por que existe um estado oculto do ambiente, que não é observado, e que é determinante para formar uma expectativa quando ao resultado daquela ação naquele contexto.

6. IMPLEMENTAÇÃO, RESULTADOS E CONCLUSÕES

Um experimento foi realizado através de simulação em computador utilizando o software desenvolvido como resultado do trabalho de [Perotto 2004b], que implementa a arquitetura básica descrita neste artigo. O universo virtual onde ocorre o experimento constitui-se de uma grade bidimensional representando um ambiente formado por paredes e espaços vazios. O agente tem capacidade de realizar três ações: andar uma célula para frente, virar-se para a esquerda

ou para a direita (nesses casos, realizando uma rotação de 90 graus). A cada instante de tempo, o agente deve executar uma das três ações. Ele anda livremente através das células vazias, mas sofre uma colisão se andar na direção de uma parede que esteja imediatamente à sua frente (figura 6).

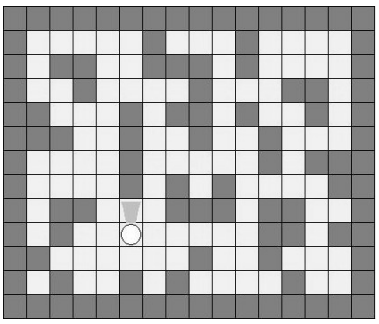
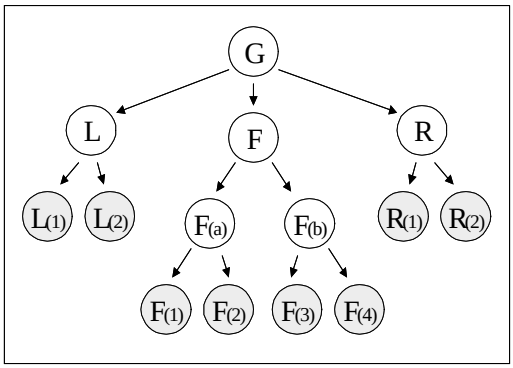


Figura 6. Ambiente da simulação: as células escuras são paredes, as células claras são espaços vazios, e o círculo branco é o agente, que está enxergando uma célula a frente.

O agente possui cinco propriedades internas: Dor, Ácido Lático, Cansaço, Exaustão e Prazer. A dor ocorre quando o agente colide com uma parede, podendo ser sentida durante um instante. O metabolismo do agente elimina-a no instante seguinte a sua ocorrência. Ácido Lático é uma variável (representando uma enzima virtual) que pode assumir valores entre 0 e 20. A cada vez que o agente anda, o ácido lático sobe um ponto, quando ele não anda, desce um ponto. Esta variável ativa outras duas: nível de cansaço, quando atinge 15 pontos, e nível de exaustão quando atinge 16 pontos. O prazer foi definido como algo que ocorre no corpo do agente quando ele caminha, e dura um instante.
O agente possui cinco percepções. Quatro delas são internas, e dizem respeito à dor, cansaço, exaustão e prazer. Note-se que a propriedade interna ácido lático não tem uma percepção interna correspondente, portanto não é diretamente percebida pelo agente. Apenas uma das cinco percepções é externa, e representa um tipo de visão. A visão permite ao agente distinguir o que está na célula imediatamente à sua frente.
O sistema emocional do agente implementa três gatilhos emocionais que têm papel avaliativo. Dois deles despertam valores emocionais negativos, e são ativados na percepção interna de dor e de exaustão. O outro gatilho emocional retorna um valor emocional positivo para a presença de prazer. Assim, o sistema emocional do agente indicará um valor avaliativo negativo para qualquer situação de dor ou exaustão, e resultará num tipo de prazer quando o agente andar.
Como o sistema cognitivo constrói expectativas para suas ações em cada contexto percebido, e incorpora o valor emocional dessas expectativas a seus esquemas, espera-se que o agente procure evitar conduzir-se a situações emocionais desagradáveis, e busque as situações agradáveis. Cognitivamente, espera-se que o agente aprenda a distinguir as situações que o levam a cada um desses estados.



	Ação	**Contexto**	**Expectativa**
L1	Esquerda	Exaustão = Não	Dor = Não Fadiga = Não Exaustão = Não Pleasure = Não
R1	Direita		
L2	Esquerda	Exaustão = Sim	Dor = Não Fadiga = Não Pleasure = Não
R2	Direita		
F1	Frente	Fadiga = Não Visão = Vazio	Exaustão = Não Dor = Não
F2	Frente	Fadiga = Não Visão = Parede	Dor = Sim Visão = Parede Exaustão = Não
F3	Frente	Fadiga = Sim Visão = Vazio	Dor = Não Exaustão = Sim
F4	Frente	Fadiga = Sim Visão = Parede	Dor = Sim Visão = Parede Exaustão = Não

Figura 7. Esquemas aprendidos pelo agente durante a simulação

Os resultados do experimento foram positivos. O agente, implementado como descrito, após alguns ciclos de interação com o ambiente (cerca de 2000), aprende as situações que o levam a diferentes estados emocionais, e, com isso, passa a evitar àquelas situações negativas, como chocar-se com uma parede, ou ficar exausto, e passa a realizar as ações que o levam a estados emocionalmente positivos. Nesse ambiente simples, o agente rapidamente converge para uma solução ótima, adotando uma conduta compatível com o valor emocional das situações, e com o comportamento esperado. Esta conduta é resultado do aprendizado das situações, que se deu através da própria experiência atuando no ambiente. É possível declarar que o agente apresentou uma autonomia no desempenho do seu comportamento, pois seus objetivos emergiram de motivações internas, e autonomia cognitiva, através da construção dos esquemas.
Os modelos construtivistas em IA mostram-se interessantes na medida em que permitem desenvolver arquiteturas de agentes inteligentes que se adaptam ao seu ambiente sem a intervenção do programador na construção de suas estruturas cognitivas. O agente, embora podendo ser programado para realizar determinadas tarefas, está livre para construir seu conhecimento na interação com o meio, e dessa forma, encontrar sozinho as soluções para os problemas que lhe surgem. Essa autonomia de construção cognitiva representa um passo importante para a IA no seu caminho em busca de agentes computacionais verdadeiramente inteligentes.

Agradecimentos

Agradecemos aos órgãos de fomento, que financiam esta pesquisa (CNPq), e também à Universidade Federal do Rio Grande do Sul, instituição que nos acolhe muito bem.

Referências

Rolf Pfeifer and Christian Scheier. *Understanding Intelligence*. MIT Press, London, England, 1999.

Franklin, Stan. Autonomous Agents as Embodied AI. Cybernetics and Systems, 28:6(1997) 499-520.

Barreto, Jorge Muniz. (1999) “Inteligência Artificial no Limiar do Século XXI”. São Paulo: UniCamp.

Belpaeme, T., Steels, L., & van Looveren, J. (1998) “The construction and acquisition of visual categories”, In: Birk, A. and Demiris, J., editors, Learning Robots, Proceedings of the EWLR-6, Lecture Notes on Artificial Intelligence 1545, Springer.

Bercht, Magda. (2001) “Em direção a agentes pedagógicos com dimensões afetivas”. Porto Alegre: UFRGS (Tese de Doutorado).

Birk, A. & Paul, W. (2000) “Schemas and Genetic Programming”, Ritter, Cruse, Dean (Eds.), Prerational Intelligence: Adaptive Behavior and Intelligent Systems without Symbols and Logic, Volume II, Studies in Cognitive Systems 36, Kluwer.

Boden, Margaret. (1979) “Piaget”, Glasgow: Fontana Paperbacks.

Coelho, Helder. (1999) “Sonho e Razão”, 2.a. ed. Lisboa: Editora Relógio d’Água.

Damasio, António. (1994) “Descartes' Error”, New York: Avon Books.

De Jong, E. D. (1999) “Autonomous Concept Formation”. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI.

Drescher, G. (1991) “Mide-Up Minds: A Constructivist Approach to Artificial Intelligence”. MIT Press.

Holland, John. (1992) “Genetic Algorithms”. Scientific American.

Inhelder B. & Cellerier G. (1996) “O desenrolar das descobertas da criança: um estudo sobre as microgêneses cognitivas”, Porto Alegre: Artes Médicas.

Kohonen, Teuvo. (1989) “Self-Organization and Associative Memory”. Berlin: Springer-Verlag.

LeDoux, J.E. (1996) “The Emotional Brain”, New York: Simon and Schuster.

McCorduck, Pamela. (1979) “Machines Who Think”, San Francisco: Freeman.

Minsky, Marvin. (1985) “The Society of Mind”. New York: Simon & Schuster.

Montangero, J. Maurice-Naville, D. (1998) “Piaget ou Inteligência em Evolução”. Porto Alegre: ArtMed.

Muñoz, M. H. S. (1999) “Proposta de Modelo Sensório Cognitivo inspirado na Teoria de Jean Piaget”. Porto Alegre: PGCC/UFRGS (Dissertação de Mestrado).

Perotto, Filipo. (2004) Proposta de um Agente Computacional Inteligente utilizando a abordagem da Inteligência Artificial Construtivista. Porto Alegre: UFRGS. (Dissertação de Mestrado, em conclusão).

Perotto, Filipo; Vicari, Rosa; Alvares, Luis Otávio. An Autonomous Intelligent Agent Architecture Based on Constructivist AI. AIAI 2004: 103-116

Perotto, Filipo. (2002) “Como é compreendida a expressão ‘Inteligência Artificial’ neste início de século XXI? – traços preliminares para uma resposta”, Porto Alegre: UFRGS (Trabalho de Diplomação).

Piaget, J. (1975a) “O Nascimento da Inteligência na Criança”, 2ª ed. Rio de Janeiro: Zahar.

Piaget, J. (1975b) “A Construção do Real na Criança”, 2ª ed. Rio de Janeiro: Zahar.

Piaget, J. (1978) “A Epistemologia Genética; Sabedoria e Ilusões da filosofia; Problemas de Psicologia Genética; Vida e Obra”, (Os Pensadores), São Paulo: Abril Cultural.

Rapp, B. (Ed.). (2001) “The Handbook of Cognitive Neuropsychology”, Hove, UK: Psychology Press.

Rocha Costa, A. C., & Dimuro, G. (2003) “Needs and Functional Foundation of Agent Autonomy”, (http://gmc.ucpel.tche.br/imqd/artigos/needs-03-04-06.pdf, 21/1/04)

Rocha Costa, A. C. (1994) “Inteligência Artificial Construtivista: princípios gerais e perspectivas de cooperação com a informática na educação”, In: Simpósio Brasileiro de Informática na Educação Anais. Porto Alegre, p. 185-198.

Rocha Costa, A. C. (1993)

Rocha Costa, A. C. (1986) “Para uma Revisão Epistemológica da Inteligência Artificial”, RP 54. Porto Alegre: CPGCC-UFRGS.

RoSloman, Aaron. (1999) “Review of Affective Computing”, AI Magazine 20 (1): 127-133.

Teixeira, João de Fernandes. (2000) “Mente, Cérebro e Cognição”. Petrópolis: Vozes.

Wazlawick, Raul. (1993) “Um Modelo Operatório Para a Construção do Conhecimento”, Florianópolis, PPGEP/UFSC (Tese de Doutoramento).

Intelligent e-learning platforms infrastructure

Marta Zorrilla
Universidad Cantabria. Spain
zorrillm@unican.es

and

Ernestina Menasalvas
Universidad Politécnica de Madrid.
Madrid, Spain
emenasalvas@fi.upm.es

and

Socorro Millán
Universidad de Valle
Cali, Colombia
millan@eisc.univalle.edu.co

Abstract

Education is an activity that is increasingly being offered through the web. E-learning highly demands technology to improve the lost 1-to-1 relationship. As most tools just provide the teacher with access summary information or primitive patterns, algorithms are being proposed for knowledge extraction as the basis for a closer teacher-student relationship. Results of data mining to be useful have to be integrated into the global e-learning infrastructure. It is shown in this paper how global webhouses especially designed are the support for all the steps of knowledge discovery process in an e-learning platform. An example is provided to illustrate the case for frequent behavior rules.

1 Introduction

Education is an activity that is increasingly being offered through the web. Students and teachers are provided with easy to use tools and are not subject to any temporal constraint. Nevertheless, e-learning environments lack a closer student-teacher relationship. The lack of this relation is evident by the fact that a teacher does not really control the evolution of his students, and students cannot express their problems and deficiencies in a natural way. This problem is found in web-based learning environments such as Virtual-U [38] and WebCT [17], including course content delivery tools, virtual workspaces for sharing resources, white boards and other tracking tools with the exception of specific tools to track and analyze student behavior and their evolution.

In spite of the fact that in e-learning environments authentications of the user is rarely a problem, the 1-to-1 relationship is not improved. It is very difficult and time consuming for teachers to thoroughly track and assess all the activities performed by all learners for all e-learning supporting tools. Moreover, it is hard to evaluate the structure of the course content and its effectiveness on the learning process.

In e-learning environments, results of application of data mining looking for a closer 1-to-1 relationship are not yet mature enough (see Section 2 for a discussion of the effort made in this direction). In [7] a framework for web intelligence processes is presented. Nonetheless, when trying to capture intelligence in e-learning environments the process has to be adapted due to the different underlying nature of data and users.

Consequently, web intelligence in e-learning environments is a challenging activity. To begin with, the very different semantics underlying the behavior of students and the different nature of the activities performed (learning and teaching), the preprocessing stage has to be adapted. In this aspect it is interesting to remark that the main change

does not only relate to gathering and storing information related to courses, topics, and activities (exams, exercises, ...) but it has also to do with parameters to estimate sessions length and behaviors that have to be redefined.

On the other hand, not only the preprocessing stage has to be adapted but the kind of patterns to be obtained which are also different from those in the case of e-commerce. Goals have to be redefined and translated into data mining. Accordingly, new data have to be captured for the goals to be fulfill. When it comes to the step of application, the behavior of the student is much more unstable than in the case of e-commerce. Consequently, some monitoring systems of changing patterns have to be deployed to secure proper interaction of the system with students and teachers. Thus, an e-learning platform acting according to user preferences is the unavoidable commitment that e-learning sponsors must face except that when doing so, complete information of courses, events, and in general any information to track user behavior cannot be neglected.

To track user navigation many approaches summarized in section 2 have been proposed. For deployment of these approaches several architectures for collaborative e-systems have been proposed. In particular, for CRM, the Gartnet Group [16] presented a three component (operational, collaborative and analytical) architecture that can be easily translated when dealing with e-learning environments. In this architecture, it is assumed that databases are used to support all the storage processes. In [47], we presented the design of a webhouse adapted to support collaborative action in a e-learning environment. Here we present how this integrated database is the support of all the processes underlying an intelligent e-learning platform. Additionally, we present some results of OLAP analysis that show how to customize the preprocessing stage in the particular case of a project being developed at Universidad de Cantabria (Spain).

The remainder of the paper is organized as follows. Section 2 presents the related work both on web intelligence in general and specifically on e-learning. In section 3, we briefly describe the full process of intelligent e-learning to highlight the relevance of a supported integrated database. Consequently, in section 4 the design of the e-learning webhouse proposed in [47] is briefly presented. Preliminary implementation results carried out at Universidad de Cantabria are presented in section 5. Section 6 presents future work and research preliminary conclusions.

2 Related work

The amazing growth of the WWW has produced an intensive research activity [10], [11]. Data mining has become an active source of study, systems, pattern discovery algorithms and techniques to be applied to WWW. In particular, web usage mining, defined by Cooley [6] as the application of data mining techniques to web clickstream data to extract web usage patterns, is at present, a common way to understand users behavior in diverse domains, e.g. business, web site design, decision support and personalization. In the web usage mining process, in which clickstream data are analyzed the preprocessing step plays an important role [7]. According to Cooley [6], [7] three steps need to be applied in order to convert raw usage data into server sessions: data cleaning, user session identification [36], [1] and page view identification. In particular, several approaches to identify user sessions have been proposed [36], [7].
Web preprocessed data are input to pattern discovery algorithms. Many algorithms have been proposed based on statistical methods , clustering , classification and association rules [28], [15], [2], [43], [12], [33], [23], [37], [13].

Results obtained by applying patterns discovery algorithms to web data can be useful to redesign a web site according to actual usage, to identify common users behavior, and to customize page views based on user profile [8], [26], [25].

On the other hand, the WWW have been used, over the past few years, for teaching and learning. In this kind of system, both students and teachers are provided with easy to use tools and are not subject to any temporal constraints. However, it is necessary to build a 1-to-1 relationship between student and teachers and provide an adequate environment to facilitate learning.

A great challenge in this context is to improve both instructional productivity and learning quality [34]. Active learning strategies have been incorporated into teaching: expanding learning experiences, taking advantage of the power of interaction and creating a dialectics between experience and dialogue [34].

Many proposals have been presented in order to contribute to improve this relationship. In Oppermann and Rashev [29] the concepts of adaptability and adaptivity to the learning systems context are discussed. Adaptable and adaptive learning systems make it possible for the student to change some parameters and to adapt to the user based on his/her needs. To make learning systems efficient to the students, adaptable and adaptive, it is necessary to adapt the learning environment according to student's goals and capabilities. Some adaptable features are related to the interface, functionality, messages and feedback. Adaptivity features are related to, for example, user level and task level.

In [18] three types of intelligent agents are introduced in order to assist teachers and students: the digital teaching assistant (DTA), the digital tutor(DT) and the digital secretary (DS). The DTA assists the teacher in teaching functions

and the DT assists students with specific learning needs. The DT could learn and become an expert in helping students and it has access to students learning profiles. Learning profiles include data about students learning credentials, learning preferences, learning styles and learning habits. Data to build profiles may include student grades and performance, students usage logs and accomplishments. The DS assists teachers and students in logistical and administrative needs. [18] also describes two major obstacles to obtain these data: the technology and software necessary to collect and analyze learning data, and the legal challenges to educational institutions for collecting student learning data.

According to Zaiane [44], [45], in spite of the significant research effort made in e-commerce systems, little has been done in web-based learning environments. Educators have very little support for evaluating student activities and for identifying different student on-line behavior. Data mining techniques could be applied to web-based distance learning in order to track student activities in a course web site to extract patterns and behavior profiles that help teachers to improve learning results.

It is necessary, in order to improve web-based learning environment, to take student behavior profiles into account. User modelling [30], [35], [3] includes individual preferences which determine the users behavior and personal information about him/her. Web usage mining and personalization approaches can also be used to build user profiles [27], [19], [24], [41], [9], [5], [39], [31], [4] in learning domains.

Machine learning techniques have been also applied to user modelling problems. In [34], IDEAL, an intelligent agent assisted system is presented. Student learning-related profiles provide students with instructional content and facilitate automatic evaluation. A unique personal agent is assigned, in IDEAL, to each student in order to manage his/her profile, including knowledge background, learning styles, interests and courses enrolled in. Bayesian belief networks are used to infer a student model from the performance data.

In [21] web log data generated by course management system are analyzed to track student learning. An information visualization technique is introduced to help instructors to understand student behavior and to become aware of what is happening in distance education classes. CourseVis, a visualization tool, generates graphical representations of student data.

Based on data extracted from log data in an education web-based system, Minaei-Bidgoli and Punch [22], use genetic algorithms in order to classify students. Using data related to educational resources (e.g. web pages, demonstrations, simulations, homework assignments, quizzes) and user information, data mining methods are analyzed for extracting knowledge to identify types of students and problems assigned to them.

An evolving e-learning system is described in [40] which can adapt itself to its users and to the open web based on the usage of its learning materials. The system users are clustered based on their learning interests.

Lei et al. [20] apply web usage mining techniques together with an analytic model in order to both study the learner interaction with content and improve instructional designs.To evaluate learning behavior, Wu and Leung [42] present preliminary results on learner behavior in which data mining techniques have been applied.

3 Process of Web Intelligence in an e-Learning system

In [14], Hu et al. propose a framework for web usage mining and business intelligence reporting which consists of four phases: data capture, webhouse construction, pattern discovery and pattern evaluation. Relating to the second stage a suitable webhouse for supporting information storage is presented.

For the particular case of e-commerce, in [16] an architecture composed by tree components is presented: operational, collaborative and analytical. On the other hand, in [32] a multiagent architecture composed of three different layers is proposed :

1. Semantic Layer containing agents related to the logic of data mining algorithms such as those used to enrich data, obtain models and refine models are presented. The dynamism obtained in algorithm deployment thanks to the agents used is remarkable.

2. Decision Layer. It is related to the agents responsible for optimizing or making decisions depending on the estimated value. This is the layer in charge of acting depending on user behavior and model applied. Prefetching and adaptive agents are described.

3. Services Provider Layer. This layer contains agents that provide several services to the rest of the agents. Services are generic and independent of the other layers.

In e-learning environments as in e-commerce for an intelligent environment to be obtained the analytical or decisional components together with the operational and collaborative components need to be integrated. For agents

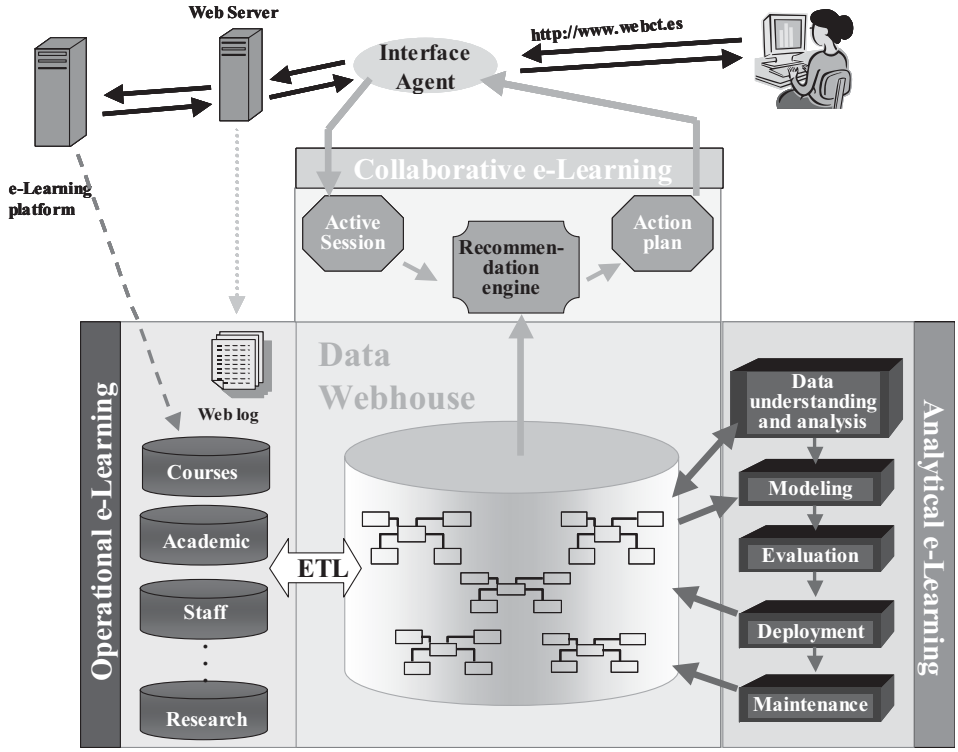


Figure 1: Web-behavior agent-based architecture layers.

or any other software to cooperate and interchange information, a database structure is required. All this situation is graphically represented in figure 1.

4 Learning Data Webhouse

Three different nature kinds of information have to be stored for proper integrated system performance:

- Information gathered by the operational systems of interest for the profiling of users: information related to students, teachers, enrollment, exams and qualifications on the one hand, and information related to courses structure and requirements on the other. This is is the information available independently of the communication channel chosen for learning to take place (face-to-face learning or e-learning).
- Navigational information: all the information related to users navigation gathered by the web server (sessions, time of stay in pages,...).
- Web site structure and content information: information about pages, lessons of the courses referenced by pages, ...
- Analysis information: Information and *knowledge* obtained by the data mining algorithms and/or other analytical tools (profiles, scores, ...).

Note that all information but the analytical prior to be stored will have to be preprocessed, transformed and integrated according to the data semantics (see figure 1).

In [47] we present a twofold webhouse structure: on the one hand, it has to store information so that intelligent algorithms can be applied and patterns obtained; on the other hand, to be the repository of patterns applied to data so it stores information about users (teachers and students) behavior and navigation patterns.

The proposed design was composed briefly of the following data marts:

- Navigational data mart to store all the information that will be needed by algorithms when calculating navigational patterns.
- Learning data mart: Students are seen as fact tables so all the history, as student and as navigator, is stored. The information in this data mart can be used to calculate incremental and refined patterns, but most importantly it can also be used to analyze those instances when the user changes behavior or exhibits certain modifications in his normal patterns.
- Teaching data mart. Similarly to the learning data mart, the teaching data mart stores information about teachers and their behavior.
- Course data mart. It is the data mart in which courses as seen as fact tables to be able to store all the information about them.

A detailed analysis of dimensions and fact tables can be found in [47]. We present a further analysis of the dimensions and facts in what follows.

5 Case-study

Since 1999, the University of Cantabria (UC), offers its students virtual courses through the WebCT platform [17]. Teachers use this environment to develop specific modules which gather parts of the subject, offer some exams or establish collaborative meetings. In two years, this offer has been extended and local students as well as students from other Spanish universities can enroll in virtual courses. A project to analyze and evaluate the use of the system and the progress and performance of the students, is being carried out since October 2004.

The case study we present in this paper is the result of pre-processing the web log file registered by Apache Web Server from October 1st, 2004 to December 31st, 2004. This file, size 1.1 Gigabytes, had initially gotten near 10.000.000 entries. After the cleaning process, which consisted of removing irrelevant log entries such as images, scripts and so on, the number of entries was reduced to 2.206.024. Before starting the process of building sessions, 3.235 learners and 322 courses were identified and finally, 48.691 sessions were built. In this last step, the duration of the visit and the number of visited resources (pages, quizzes, forums, ...) were calculated. Besides, the visited pages during each session were registered to be used in the incoming navigational behavior analysis.

In what follows we briefly present some of the measures already obtained that, as it can be easily observed, make the preprocessing stage a lot easier.

5.1 Usage pattern analysis

The session analysis allows us to understand how the system was used, how the course structure was designed and how the learners' behavior evolved over time.

Figures 2 and 3 show sessions per learner, average session time and pages per session measured. Figure 2 shows the results obtained when all courses in this period (in weeks) are taking into account. In figure 3 results are shown by the most frequently enrolled in and well-designed course. This analysis is possible thanks to the data warehouse.

In Figure 2, it can be observed that the average connection time by students, in this term is low, less than 15 minutes, though in Figure 3, time increased to 20 or 30 minutes. This suggests to us that when preparing sessions for data mining purposes these parameters have to be taken into account. Reduced connection time may indicate that students are not using collaborative tools (e.g. chats, mail) very often. This pattern will have to be further analyzed.

5.2 Learner distribution over time

Other interesting information that the data warehouse makes possible for us to analyze, consists of analyzing the distribution over time of the number of connected students depending on the week day and/or time of the day. An example of this information is shown in Figure 4. In this figure, one can also observe whether the connection was made inside (value 1) or outside (value 0) the UC campus.

The results described show just part of the information that can be analyzed and stored in an integrated e-learning platform. Only by having such an integrated supporting data warehouse will it be possible to have intelligent e-learning platforms.

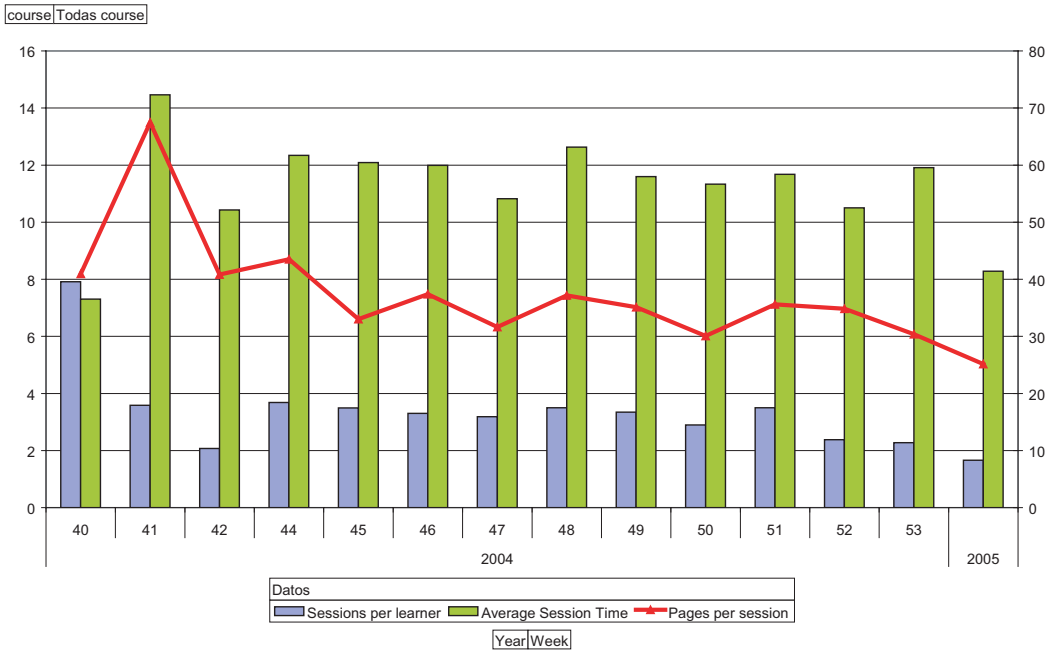


Figure 2: Usage pattern analysis of the full system.

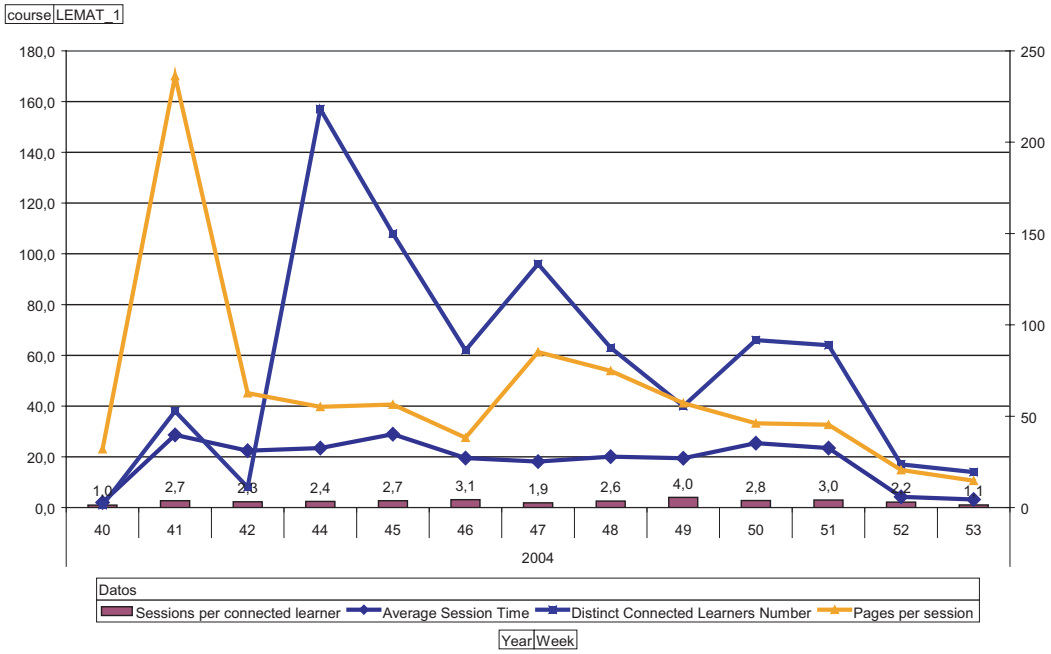


Figure 3: Usage pattern analysis of a well-designed course.

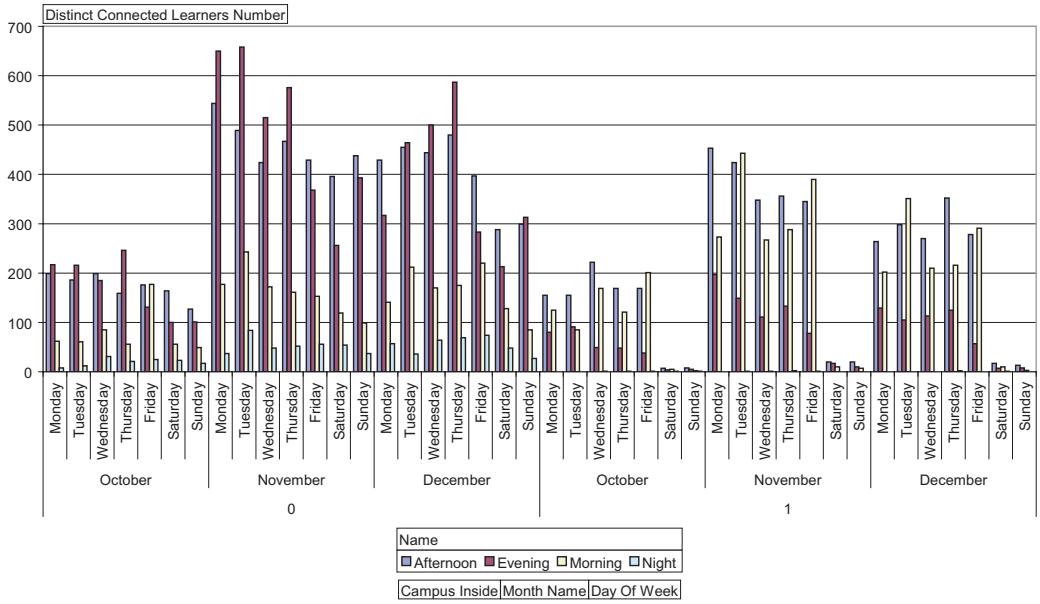


Figure 4: Learner distribution over time

6 Conclusion and outlook

Intelligent e-learning tools mean having the ability to understand and profit from experience. Intelligence support is made up of different components that have to cooperate in an integrated way. Although the results of applying data mining to e-learning environments are not yet mature enough, the experience from related internet examples, e-commerce is a clear example, shows that for an e-commerce system to act intelligently, algorithms alone are not enough. A supporting database is needed so that, for example, agents in an agent-based architecture can cooperate and/or interchange information.

A prototype of that design has been already implemented and tested as a supporting data warehouse for analysis at Universidad de Cantabria (see [46]) and the results shown in section 5 are promising. In fact, the results of the OLAP analysis show how before making preprocessing and calculating sessions, values of user navigation have to be taken into account. Otherwise, patterns can be misleading. The parameter used for session establishment can be now selected depending on courses and kinds of users.

Our next step is to adapt the general design presented here to establish well-known discovery processes in e-learning. Thus we are working on adapting this design as an automated incremental repository for typologies of students. In this sense, information needed for Markov models application in student behavior is our next goal.

7 Acknowlegement

This work has been partially supported by Spanish Ministry of Education under project ”TIN2004-05873” and project ”TIC-2002-01306”

References

[1] Bettina Berendt, Bamshad Mobasher, Miki Nakagawa, and Myra Spiliopoulou. The impact of site structure and user environment on session reconstruction in web usage analysis. In *In O. Zaane, J. Srivastava, M. Spiliopoulou,*

B. Masand (Eds.), WEBKDD 2002 - Mining Web Data for Discovering Usage Patterns and Profiles. LNAI 2703. Berlin: Springer. [http://www.wiwi.hu-berlin.de/ berendt/], pages 159–179, 2002.

[2] J. Borges. A data mining model to capture userweb navigation patterns. In *Ph.D. Thesis, Department of Computer Science, University College London, 2000. [http://paginas.fe.up.pt/ jlborges/]*, 2000.

[3] Peter Brusilovsky. Knowledge tree: A distributed architecture for adaptive e-learning. In *Proccedings of WWW04, May 17-22, New York*, pages 104–113, 2004.

[4] Ricardo Carreira, Jaime Crato, Daniel Goncalves, and Joaquim A Jorge. Evaluating adaptive user profiles for news classification. In *Proccedings of IUI04, January 13-16, Madeira, Portugal*, pages 206–212, 2004.

[5] Yoon Ho Cho, Jae Kyeong Kim, and Soung Hie Kim. A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, 23:329–342, 2002.

[6] Robert Cooley. Web usage mining: Discovery and application of interesting patterns from web data. In *Ph.D. Thesis. University of Minnesota, 2000 [http://www-users.cs.umn.edu/ cooley/]*, 2000.

[7] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. In *Knowledge and Information Systems*, volume 1 (1), 1999.

[8] Honghua Dai and Bamshad Mobasher. A road map to more effective web personalization: Integrating domain knowledge with web usage mining. In *Proceedings of the International Conference on Internet Computing 2003 (IC'03), Las Vegas, Nevada, June 2003. [http://maya.cs.depaul.edu/ mobasher/pubs.html]*, 2003.

[9] Magdalini Eirinaki, Charalampos Lampos, Stratos Paulakis, and Michalis Vazirgiannis. Web personalization integrating content semantics and navigational patterns. In *Proccedings of WIDM04, November 12-13, Washington*, pages 72–79, 2004.

[10] Federico Michele Facca and Pier Luca Lanzy. Mining interesting knowledge from weblogs: a survey. *Data and Knowledge Engineering*, 53:225–241, 2005.

[11] J. Han and M. Kamber. *Data Mining Concepts and Techniques*. Morgan Kaufmann , Mar 2001.

[12] B. Hay, G. Wets, and K. Vanhoof. Clustering navigation patterns on a website using a sequence alignment method. In *Intelligent Techniques for Web Personalization: IJCAI 2001, 17th Int. Joint Conf. on Artificial Intelligence, August 4, 2001, Seattle, WA, USA*, pages 1–6, 2001.

[13] J. Heer and E.H. Chi. Mining the structure of user activity using cluster stability. In *Proceedings of the Workshop on Web Analytics, Second SIAM Conference on Data Mining, ACM Press*, 2002.

[14] Xiaohua Hu and Nick Cercone. An olam framework for web usage mining and business intelligence reporting. In *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems*, volume 2 (7), pages 950–955, 2002.

[15] A. Huang and N. Cercone. Comparison of interestingness functions for learning web usage patterns. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management, ACM Press, 2002*, pages 617–620, 2002.

[16] Elizabeth Shahnam. META Group Inc. The customer relationship management ecosystem.

[17] WebCT Inc. Webct. http://www.webct.com/.

[18] Ali Jafari. Conceptualizing intelligent agents for teaching and learning. *EDUCAUSE QUARTERLY*, 3:28–34, 2002.

[19] Soren Jespersen, Torben Bach Pedersen, and Jesper Thorhauge. Evaluating the markov assupmtion for web usage mining. In *Proccedings of WIDM03, November 7-8, New Orleans*, pages 82–89, 2003.

[20] Xu Lei, Claus Pahl, and Dave Donnellan. An evaluation techniques for content interaction in web-based teaching and learning environments. In *Proccedings of The 3rd IEEE International Conference on Advanced Learning Technologies*, 2003.

[21] Riccardo Mazza and Vania Dimitrova. Visualising student tracking data to support instructors in web-based distance education. In *Proceedings of WWW 2004, May 17-22, 2004, Nwe York.*, pages 154–160, 2004.

[22] Behrouz Minae-Bigdoli and William F. Punch III. Using genetic algorithm for data mining optimzation in an educational web-based system. In *Proccedings of WWW 2004, May 17-22, New York*, pages 675–684, 2004.

[23] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *In Proceedings of the 3rd ACM Workshop on Web Information and Data Management (WIDM01), held in conjunction with the International Conference on Information and Knowledge Management (CIKM 2001), Atlanta, Georgia, November 2001*, page [http://maya.cs.depaul.edu/ mobasher/pubs.html], 2001.

[24] Bamshad Mobasher. Webpersonalizer: A server-side recommender system based on web usage mining. In *the Proceedings of the 9th Workshop on Information Technologies and Systems (WITS'99), Charlotte, NC, Dec. 1999. [http://maya.cs.depaul.edu/ mobasher/pubs.html]*, 1999.

[25] Bamshad Mobasher. *Practical Handbook of Internet Computing. Munindar P. Singh (Ed.) [http://maya.cs.depaul.edu/ mobasher/pubs.html]*, chapter Web Usage Mining and Personalization. CRC Press, 2005.

[26] Bamshad Mobasher, Hoghua Dai, Tao Luo, Yuqing Sun, and Jiang Zhu. Integrating web usage and content mining for more effective personalization. In *Proceedings of the International Conference on E-Commerce and Web Technologies (ECWeb2000), September 2000, Greenwich, UK. [http://maya.cs.depaul.edu/ mobasher/pubs.html]*, 2000.

[27] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. In *In Data Mining and Knowledge Discovery, Kluwer Publishing Vol. 6, No. 1. [http://maya.cs.depaul.edu/ mobasher/pubs.html]*, pages 61–82, 2002.

[28] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. Exploiting web log mining for web cache enhancement. *Lecture Notes in Computer Science*, 2356:68–87, 2002.

[29] Reinhard Oppermann and Rossen Rashev. Adaptability and adaptivity in learning systems. *In: A. Behrooz (ed.): Knowledge Transfer (Vol. II). Proceedings on Knowledge Transfer, July 14 - 16, 1997 in London, UK*, 3:173 – 179, 1997.

[30] Christos Papatheodorou. Machine learning in user modeling. *Lecture Notes in Artificial Intelligence (LNAI,) Springer-Verlag, 2001*, 2049:286–294, 2001.

[31] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.

[32] Ernestina Menasalvas Ruiz, B. Pardo, Socorro Millán, Esther Hochsztain, and José M. Peña Sánchez. Expected value of user sessions: Limitations to the non-semantic approach. In *Web Intelligence*, pages 562–565, 2003.

[33] C. Shahabi and Y.S. Chen. Improving user profiles for e-commerce by genetic algorithms. In *E-Commerce and Intelligent Methods Studies in Fuzziness and Soft Computing, Vol. 105, VIII, J. Segovia, P.S. Szczepaniak, M. Niedzwiedzinski (Eds.), 2002*, 2002.

[34] Yi shang, Hongchi Shi, and Su-Shing Chen. An intelligent distributed environment for active learnig. In *Proceedings of WWW01, May 1-5, 2001, Hong Kong. [http://www10.org/cdrom/papers/207/WWW10-207.html]*, 2001.

[35] Goran Simic, Dragan Gasevic, and Vladan Devedzic. Semantic web and intelligent learning management systems. In *Proceedings of ITS 2004: International Conference on Intelligent Tutoring Systems. 30 August 3 September 2004, Macei-Alagoas, Brazil*, 2004.

[36] M. Spiliopoulou, B. Mobasher, B. Berendt, and M. Nakagawa. A framework for the evaluation of session reconstruction heuristics in web-usage analysis. In *INFORMS Journal on Computing*, volume 15 (7), pages 171–190, 2003.

[37] J. Srivastava, R. Cooley, M. Deshpande, and P.N. Tan. Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explorations 1 (2)*, 1:12–23, 2000.

[38] Denise Stockley, Chris Groeneboer, Tom Calvert, and Linda Harasim. Virtual-u: An online learning environment. In *WebNet*, 1997.

[39] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proccedings of WWW 2004, May 17-22, New York*, pages 675–684, 2004.

[40] Tiffany Ya Tang and Gordon McCalla. Smart recommendation for an evolving e-learning system. In *11th International Conference on Artificial Intelligence in Education (AIED'2003). Workshop on Technologies for Electronic Documents for Supporting Learning. July 20-24, Sydney, Australia. 2003*, pages 699–710, 2003.

[41] Ahmad M. Ahmad Wasfi. Collecting user access patterns for building user profiles and collaborative filtering. In *IUI 99, Redondo Beach*, pages 57–64, 1999.

[42] Albert K W Wu and C H Leung. Evaluating learning behavior of web-based training (wbt) using web log. In *Proccedings of the IEEE International Conference on Computers in Education*, 2002.

[43] Y. Xie and V.V. Phoha. Web user clustering from access log using belief function. In *Proceedings of the First International Conference on Knowledge Capture 2001, ACM Press*, pages 202–208, 2001.

[44] Osmar Zaiane. Web usage mining for a better web-based learning environment. In *Proc. of Conference on Advantage Technology for Education. Alberta, Canada. 2001.*, 2001.

[45] Osmar Zaiane and Jun Luo. Towards evaluating learners' behaviour in a web-based distance learning environment. In *Proc. of IEEE International Conference on Advanced Learning Technologies (ICALT01) Madison, WI, August 6-8, 2001.*, pages 357–360, 2001.

[46] M. E. Zorrilla, E. Menasalvas, D. Marn, E. Mora, and J. Segovia. Web usage mining project for improving web-based learning sites. In *Procedings of the Ninth International Conference on Computer Aided Systems Theory,Las Palmas de Gran Canaria, Spain*, 2005.

[47] Marta Zorrilla, Socorro Millán, and Ernestina Menasalvas. Data webhouse to support web intelligence in e-learning environments. In *Submitted to IEEE International Conference on Granular Computing. Beijing, China*, 2005.

Uma Proposta para a Construção de *Ensembles* Simbólicos que Explicam suas Decisões

Flávia Cristina Bernardini*e Maria Carolina Monard
Laboratório de Inteligência Computacional
Instituto de Ciências Matemáticas e Computação
Universidade de São Paulo
Av. do Trabalhador Sancarlense, 400 – Caixa Postal 668
CEP 13560-970 São Carlos, SP
{fbernard,mcmonard}@icmc.usp.br

Abstract

Data Mining applications generally use learning algorithms in order to induce knowledge. To accomplish this task, these algorithms should be able to operate with massive data sets. Several techniques, such as data sampling, can be used to scale up learning algorithms to deal with large datasets. In order to induce classifiers, learning algorithms can thus be applied to smaller samples of the original dataset. The individual classifiers can then be combined into ensembles which under certain conditions can be more accurate than the individual classifiers. However, when compared to symbolic classifiers, ensembles often lack the facility to explain their decisions. This work explores a method to offer concise explanation of ensembles decisions whenever the ensembles are composed by a combination of symbolic classifiers. Different methods used to construct ensembles are also described.

Keywords: Symbolic Machine Learning, Symbolic Ensembles, Combining Classifiers.

Resumo

Em aplicações práticas de Data Mining geralmente são utilizados algoritmos de aprendizado para induzir conhecimento. Entretanto, esses algoritmos nem sempre estão preparados para trabalhar com grandes bases de dados. Uma maneira de possibilitar tais algoritmos a trabalhar com grandes bases é utilizar a abordagem de *ensembles* de classificadores associada a técnicas de amostragem de conjuntos de dados. Para induzir os classificadores, algoritmos de aprendizado podem ser aplicados em amostras do conjunto de dados original. Os classificadores induzidos podem ser, então, combinados em *ensembles*, os quais podem ser, sob certas condições, mais precisos que os classificadores individuais que o compõem. Entretanto, *ensembles* freqüentemente não conseguem explicar suas decisões de classificação. Neste trabalho exploramos um método para oferecer explicações concisas das decisões de classificações de *ensembles* simbólicos e também apresentamos métodos para construir esses *ensembles*.

Palavras chave: Aprendizado de Máquina Simbólico, *Ensembles* Simbólicos, Combinação de Classificadores.

1 INTRODUÇÃO

Em Aprendizado de Máquina — AM — supervisionado, além da indução de classificadores precisos, é também importante analisar o conhecimento induzido com o objetivo de descobrir novo conhecimento. Nesse caso, há um forte apelo para usar algoritmos de AM simbólicos, nos quais o modelo induzido pode ser diretamente interpretado pelo usuário/especialista do domínio. Porém, quando são utilizadas grandes bases (conjuntos) de dados, muitas vezes os algoritmos de AM simbólicos disponíveis não estão prontos para lidar com tais bases. Ainda, mesmo quando os conjuntos de dados sejam suportados por tais algoritmos, nem sempre é

*Trabalho realizado com auxílio da FAPESP, Brasil, Proc. Nº 02/06914-5.

possível induzir um classificador suficientemente preciso, ou seja, que classifique novos casos com a precisão exigida pelo especialista. Uma possível solução é utilizar a abordagem de *ensembles* de classificadores, para tornar os algoritmos de AM disponíveis capazes de lidar com grandes bases e/ou melhorar o poder preditivo da classificação. Um *ensemble* consiste de um conjunto de classificadores cujas decisões individuais são combinadas de alguma maneira para classificar novos casos [9, 11].

As principais abordagens de *ensembles* encontradas na literatura apresentam comportamento tipo caixa preta, ou seja, não têm condições de oferecer uma explicação que justifique a classificação de novos exemplos [6, 12, 2, 7]. Em algumas situações, essas explicações são obrigatórias. Por exemplo, em certos países, ao informar ao cliente de um banco se seu crédito foi aprovado ou não, existe a necessidade legal de se justificar o motivo de sua aprovação ou negação. Já os *ensembles* construídos com a abordagem de construção de *ensembles* utilizada neste trabalho fornecem, para cada exemplo, a classificação desse exemplo e uma explicação relacionada à classificação realizada. Entretanto, muitas vezes essa explicação pode ser muito extensa para ser fornecida ao usuário do *ensemble* e/ou ao especialista do domínio, pois essa explicação está relacionada com as regras disparadas pelo conjunto de classificadores que constituem o *ensemble*.

O objetivo deste trabalho é propor métodos para construir *ensembles* de classificadores simbólicos bem como propor um método para resumir a explicação fornecida pelos *ensembles* construídos, apresentando assim, ao usuário do *ensemble*, uma explicação mais concisa e compreensível, relacionada à classificação de novos casos (exemplos).

O restante deste artigo está organizado como segue: na Seção 2 são descritos outros trabalhos relacionados à construção de *ensembles*; na Seção 3 são apresentados alguns conceitos para possibilitar a compreensão deste trabalho bem como a notação utilizada; na Seção 4 são descritos os métodos propostos neste trabalho para a construção de *ensembles* simbólicos; na Seção 5 é mostrado o método utilizado para resumir a explicação fornecida ao usuário pelos *ensembles* simbólicos na classificação de novos exemplos; na Seção 6 são descritos os experimentos realizados utilizando um conjunto de dados naturais e os resultados obtidos; por fim, na Seção 7 encontram-se as conclusões deste trabalho e trabalhos futuros.

2 TRABALHOS RELACIONADOS

Ensembles de classificadores têm sido bastante estudados nos últimos anos na área de AM, com a finalidade de melhorar o poder de predição de algoritmos de aprendizado supervisionado. A tarefa de construção de *ensembles* de classificadores pode ser dividida em duas sub-tarefas [9]. A primeira consiste em induzir um conjunto de classificadores componentes do *ensemble*. A segunda tarefa consiste em decidir como combinar as decisões dos classificadores componentes para classificar novos exemplos. Alguns trabalhos propõem métodos de construção de *ensembles* — *bagging* [6], *boosting* [12], dentre outros — e variações desses métodos — por exemplo, *wagging* [2] e *arcing* [7] — que envolvem ambas sub-tarefas de construção de *ensembles*. Vários trabalhos utilizam um grande número de classificadores para compor um *ensemble* [10, 2] com o objetivo de melhorar a sua precisão. Porém, em [3], é feita uma análise que explica ser possível construir bons *ensembles* de classificadores com um pequeno número de classificadores componentes. Além disso, trabalhos mais recentes têm mostrado maior preocupação com a fase de combinação de classificadores [15].

Para realizar a combinação, existem propostas de métodos de *stacking* [19], nos quais é construído um classificador para tomar a decisão final do *ensemble* dadas as decisões individuais dos classificadores. Em ambos trabalhos [15, 19], resultados satisfatórios são apresentados e é mostrado que não é necessário utilizar métodos específicos para a construção dos classificadores que compõem o *ensemble*, como os utilizados nos trabalhos iniciais em *ensembles* de classificadores [6, 12].

Os métodos de construção de *ensembles* utilizados nesses trabalhos apresentam comportamento do tipo caixa preta, ou seja, não permitem explicar as decisões de classificação tomadas pelos *ensembles* construídos. Entretanto, em algumas situações, é obrigatório o fornecimento de explicação de classificações, como citado anteriormente. Porém, não é de nosso conhecimento trabalhos relacionados com o tratamento de explicação de decisões tomadas por *ensembles* de classificadores simbólicos, o qual é o foco deste trabalho.

3 CONCEITOS E NOTAÇÃO

No problema padrão de AM supervisionado, ao algoritmo de aprendizado de máquina é dado um conjunto de exemplos de treinamento S com N exemplos $T_i, i = 1, ..., N$, escolhidos de um domínio $\mathcal{X}$ com uma

distribuição D fixa, desconhecida e arbitrária, da forma $\{(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\}$ para alguma função desconhecida $y = f(\mathbf{x})$. Os $\mathbf{x}_i$ são tipicamente vetores da forma $(x_{i1},x_{i2},...,x_{iM})$, com valores discretos ou contínuos, onde x_{ij} refere-se ao valor do atributo j, denominado X_j, do exemplo T_i. Os valores y_i referem-se ao valor do atributo Y, freqüentemente denominado classe — Tabela 1.

	X_1	X_2	...	X_M	Y
T_1	x_{11}	x_{12}	...	x_{1M}	y_1
T_2	x_{21}	x_{22}	...	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_N	x_{N1}	x_{N2}	...	x_{NM}	y_N

Tabela 1: Conjunto de exemplos no formato atributo-valor

Em problemas de classificação, tratados neste trabalho, o atributo classe y_i é discreto, ou seja, $y_i \in \{C_1, C_2, ..., C_{N_{Cl}}\}$. A partir do conjunto de treinamento S, um *classificador* $\mathbf{h}$ é induzido. No caso de aprendizado simbólico proposicional, $\mathbf{h}$ pode ser transformado em um conjunto de regras **if-then**, não ordenadas ou disjuntas, tal que $\mathbf{h} = \{R_1, R_2, ..., R_{N_R}\}$. Neste trabalho, as regras que participam de $\mathbf{h}$ são regras de classificação, definidas mais adiante, ainda nesta seção.

Um complexo é uma disjunção de conjunções de testes de atributos da forma X_i *op Valor*, onde X_i é o nome do atributo, *op* é um operador pertencente ao conjunto $\{=, \neq, <, \leq, >, \geq\}$ e *Valor* é um valor válido para o atributo X_i. Uma *regra proposicional* R apresenta a forma **if** B **then** H ou, simbolicamente, $B \rightarrow H$, onde H é a *cabeça*, ou a conclusão da regra R, e B é o *corpo*, ou condição de R. H e B são ambos complexos sem atributos em comum.

A *cobertura* de uma regra $R = B \rightarrow H$ é definida como segue: exemplos que satisfazem B (o corpo da regra) compõem o conjunto de cobertura de R; em outras palavras, esses exemplos são cobertos por R. Uma *regra de classificação* assume a forma **if** B **then** classe $= C_i$. Ou seja, a cabeça H de uma regra de classificação é classe $= C_i$, com $C_i \in \{C_1, ..., C_{N_{C_l}}\}$. A maioria dos algoritmos de indução de regras de classificação pertencem a uma das famílias de algoritmos: "separar-para-conquistar" e "dividir-para-conquistar". Algoritmos da primeira família geralmente usam um algoritmo iterativo e guloso de cobertura sobre o conjunto de exemplos de treinamento. A cada iteração, o algoritmo encontra a melhor regra e remove os exemplos cobertos por essa regra. Finalmente, um classificador é construído com as regras encontradas. Este classificador é composto por uma lista de regras ordenadas (ou lista de decisão), caso tiverem sido removidos, em cada iteração do algoritmo, todos os exemplos cobertos pela regra descoberta, ou é composto por um conjunto de regras não ordenadas, caso tiverem sido removidos, em cada iteração do algoritmo, somente os exemplos corretamente cobertos pela regra. Algoritmos da segunda família — "dividir-para-conquistar" — constroem um classificador global utilizando uma estratégia *top-down*, refinando, a cada iteração, a teoria parcial. Geralmente, classificadores induzidos por algoritmos dessa família são expressos como árvores de decisão, as quais podem ser escritas como um conjunto de regras disjuntas e não ordenadas.

Dadas duas regras de classificação $R_i = B_i \rightarrow H_i$ e $R_j = B_j \rightarrow H_j$ com cabeças iguais ($H_i = H_j$), é muito simples determinar se R_i é uma generalização de R_j, *i.e* R_i *subsume* R_j, ou se R_i é uma especialização de R_j, *i.e* R_j *subsume* R_i, considerando o conjunto de testes de atributos que participam do corpo dessas regras. Sejam A_i e A_j, respectivamente, os conjuntos de testes de atributos que participam nos corpos B_i e B_j das regras R_i e R_j respectivamente. Então, R_i é uma generalização de R_j, ou R_j é uma especialização de R_i se $A_i \subseteq A_j$. Por exemplo, considerando as duas regras de classificação:

$R_i =$ **If** $at_1 = 2$ **and** $at_2 > 4$ **then** *class* $= +$
e
$R_j =$ **If** $at_1 = 2$ **and** $at_2 > 4$ **and** $at_3 = 5$ **then** *class* $= +$

então

$A_i = \{at_1 = 2, at_2 > 4\}$
e
$A_j = \{at_1 = 2, at_2 > 4, at_3 = 5\}$.

Assim, R_i é uma generalização de R_j pois $A_i \subseteq A_j$[1]. Com a finalidade de simplificar uma explicação composta por um conjunto de regras $\mathbf{R}_{expl}$, a explicação continua sendo válida removendo-se de $\mathbf{R}_{expl} = \{R_1, ..., R_P\}$ as regras que são especialização de outras regras também pertencentes a R_{expl} porque, dadas duas regras R_i e R_j, e dado que R_i é uma generalização de R_j, os exemplos cobertos por R_j são cobertos pela regra mais geral R_i. Essa idéia é utilizada neste trabalho para simplificar a explicação de *ensembles* simbólicos, como explicado na Seção 5.

4 COMBINAÇÃO DE CLASSIFICADORES SIMBÓLICOS

Como mencionado, o processo de construção de um *ensemble* de classificadores pode ser dividido em duas sub-tarefas [11]:

1. indução de um conjunto de classificadores; e
2. combinação das decisões dos classificadores induzidos para classificar novos exemplos.

Neste trabalho, a primeira tarefa é realizada da maneira usual: dados o número de classificadores a serem induzidos, L, e um conjunto de dados (exemplos), S, primeiramente são retiradas L amostras $S_1, ..., S_L$ de S com ou sem reposição. Cada uma dessas amostras, *i.e* conjunto de exemplos de treinamento, é usada como entrada para um algoritmo de aprendizado simbólico para se induzir L hipóteses (classificadores) denominadas $\mathbf{h}_1, ..., \mathbf{h}_L$. Deve ser observado que o algoritmo não precisa ser o mesmo para todas as L amostras, qualquer algoritmo de aprendizado simbólico pode ser utilizado. A única restrição é quanto ao tipo de regras geradas pelo algoritmo, as quais devem ser regras não ordenadas, descritas na Seção 3. Após, dado um novo exemplo $\mathbf{x}$ a ser classificado, as decisões individuais do conjunto de L hipóteses $\{\mathbf{h}_1, ..., \mathbf{h}_L\}$ devem ser combinadas para gerar a classificação final. Na Figura 1 é ilustrado o método proposto, onde $Combinar(\mathbf{h}_1(\mathbf{x}), ..., \mathbf{h}_L(\mathbf{x}))$ constitui o *ensemble* simbólico.

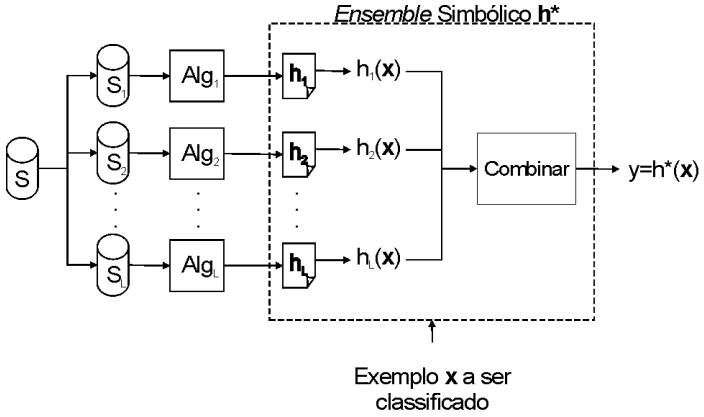


Figura 1: Um método para construção de *ensembles* de classificadores.

Para combinar as classificações (decisões) realizadas pelos classificadores individuais, são propostos os seguintes 3 (três) métodos para realizar a combinação:

1. ***Unweighted Voting*** − **UV**: $\mathbf{x}$ é rotulado com a classe que receber mais votos dos L classificadores;
2. ***Weighted by Mean Voting*** − **WMV**: o número de votos na classe dada por cada classificador $\mathbf{h}_l$ ao classificar $\mathbf{x}$ é ponderado pela média estimada de erro do classificador, representado por $m\_err(\mathbf{h}_l)$, e $\mathbf{x}$ é então rotulado com a classe que tiver o peso máximo dos L classificadores:

[1]Deve ser observado que $A_i \subseteq A_j$ considera que para testes de um mesmo atributo envolve análise de subconjuntos quando o atributo X_i *op Valor*, o operador "$\leq$" é uma generalização de "$<$", "$\geq$" é uma generalização de "$>$", e assim sucessivamente.

$$WMV(\mathbf{x}, C_v) = \max_{C_i \in \{C_1,\ldots,C_{N_{Cl}}\}} \sum_{l=1}^{L} g(\mathbf{h}_l(\mathbf{x}), C_i) \text{ , onde}$$

$$g(\mathbf{h}_l(\mathbf{x}), C_i) = \begin{cases} \lg((1 - m\_err(\mathbf{h}_l))/m\_err(\mathbf{h}_l)) & \text{se } \mathbf{h}_l(\mathbf{x}) = C_i, \\ 0 & \text{caso contrário.} \end{cases}$$

3. ***Weighted by Mean and Standard Error Voting*** – **WMSV**: similar ao método anterior, mas considerando também o erro padrão da média do erro do classificador, representado por $se\_err(\mathbf{h}_l)$), para calcular o peso do classificador correspondente:

$$WMSV(\mathbf{x}, C_v) = \max_{C_i \in \{C_1,\ldots,C_{N_{Cl}}\}} \sum_{l=1}^{L} g(\mathbf{h}_l(\mathbf{x}), C_i) \text{ , onde}$$

$$g(\mathbf{h}_l(\mathbf{x}), C_i) = \begin{cases} \lg((1 - m\_err(\mathbf{h}_l))/m\_err(\mathbf{h}_l)) \\ \quad + \lg((1 - se\_err(\mathbf{h}_l))/se\_err(\mathbf{h}_l)) & \text{se } \mathbf{h}_l(\mathbf{x}) = C_i, \\ 0 & \text{caso contrário.} \end{cases}$$

Com o objetivo de avaliar essas propostas, foi implementado um sistema computacional denominado Ensemble Learning Environment (ELE), integrado ao ambiente computacional Discover. O ambiente Discover tem como principal objetivo integrar e padronizar os diversos projetos desenvolvidos no Laboratório de Inteligência Computacional – LABIC[2] – do ICMC-USP, relacionados com pré-processamento de conjuntos de dados, aquisição automática de conhecimento e avaliação de conhecimento [17]. Na maioria desses projetos, incluindo o projeto ELE, diversas tarefas tais como transformação de dados e formatos, execução de algoritmos, medições, entre outras, devem ser executadas diversas vezes. Assim, muitas ferramentas foram e estão sendo implementadas em *Perl* [20] no Discover, como bibliotecas de classes, para automatizar parcial ou integralmente algumas dessas tarefas. O ambiente Discover oferece vantagens em relação a outros sistemas com objetivos semelhantes, pois permite a visão unificada que os formatos baseados em padrões proporcionam ao pesquisador (desenvolvedor) de novos componentes. Os padrões de representação foram sendo definidos por área, sendo que em [16] é proposta uma sintaxe padrão para representação de conhecimento de diversos indutores simbólicos denominada $\mathcal{PBM}$. Para a representação de dados foi proposta uma sintaxe padrão, denominada DSX – *Discover Dataset Sintax*, a qual permite a utilização da biblioteca de classes DOL [1], para converter os arquivos de dados para sintaxe utilizada por diversos sistemas de aprendizado simbólico, tais como $\mathcal{C}4.5$, $\mathcal{C}4.5rules$, $\mathcal{CN}2$ entre outros. O sistema ELE, no qual estão implementados os métodos de combinação de classificadores UV, WMV e WMSV, foi desenvolvido como uma biblioteca de classes do Discover utilizando várias de suas funcionalidades.

Em [4], são descritos experimentos realizados com o sistema ELE utilizando o conjunto de dados Nursery, da UCI [13]. Os resultados obtidos foram satisfatórios, considerando que as estimativas de erro dos *ensembles* de classificadores foram menores que as estimativas de erro dos classificadores que o compõem. Entretanto, como mencionado anteriormente, além da preocupação com o poder de predição em problemas de AM em termos de precisão de classificação, existe também a preocupação em fornecer ao usuário a explicação da predição feita pelo sistema. Neste trabalho, são apresentados os resultados obtidos nos experimentos realizados com o conjunto de dados Nursery, tanto em relação às estimativas de erro dos *ensembles* — os quais encontram-se melhor detalhados em [4] — quanto em relação à explicação relacionada com a classificação dada pelo *ensemble*, foco deste trabalho. Na próxima seção é descrito um algoritmo que resume as regras fornecidas pelo *ensemble* utilizado para predizer um dado exemplo, de maneira a mostrar ao usuário uma explicação mais concisa e, portanto, mais fácil de ser compreendida pelo usuário do sistema.

5 CONSTRUÇÃO DA EXPLICAÇÃO

Dado um *ensemble* de classificadores simbólicos e um exemplo $\mathbf{x}$ a ser classificado, o sistema ELE pode utilizar uma das 3 (três) maneiras descritas na seção anterior para combinar as decisões de cada um dos

[2] http://labic.icmc.usp.br

classificadores componentes. Como os classificadores são simbólicos, o *ensemble* oferece como saída, além da classificação final do exemplo $\mathbf{x}$, todas as regras que cobrem esse exemplo. Dessas regras, são selecionadas aquelas cuja cabeça (classe) H_i prediz para o exemplo $\mathbf{x}$ a mesma classe que o *ensemble*, as quais formam o conjunto de regras $\mathbf{R}_{expl}$. Entretanto, nesse conjunto podem existir regras mais gerais que outras e, como mencionado na Seção 3, para tornar a explicação mais simples, retira-se de $\mathbf{R}_{expl}$ as regras mais especializadas em relação a outras também pertencentes a $\mathbf{R}_{expl}$. Ou seja, analisando as regras em $\mathbf{R}_{expl}$ duas a duas, ficam no conjunto final $\mathbf{R}'_{expl}$ somente as regras que subsumem outras regras.

O procedimento para encontrar a explicação final $\mathbf{R}'_{expl}$ é o seguinte: o corpo de cada regra R_e pertencente a $\mathbf{R}_{expl}$, B_e, consiste de uma conjunção de testes de atributos; cada corpo, então, é transformado no conjunto correspondente A_e de testes de atributos. O conjunto composto por esses conjuntos A_e é denominado $\mathbf{A}_{expl}$. Após construído o conjunto $\mathbf{A}_{expl}$, o Algoritmo 1 descreve como é construída a explicação mais geral da classificação atribuída pelo *ensemble* ao exemplo $\mathbf{x}$, a qual será mostrada ao usuário.

Algoritmo 1 Simplificação de Explicação

Require: $\mathbf{x}$: exemplo classificado pelo *ensemble*;
$\mathbf{R}_{expl} = \{\mathbf{R}_i, ..., \mathbf{R}_j, ..., \mathbf{R}_q\}$: conjunto de regras dos classificadores $\mathbf{h}_1, ..., \mathbf{h}_L$ que constituem o *ensemble*, cobrem $\mathbf{x}$ e predizem a classe $\mathbf{h}^*(\mathbf{x})$.

1: **procedure** explicacao($\mathbf{x}$,$\mathbf{R}_{expl}$,$\mathbf{A}_{expl}$)
2: $\mathbf{A}_{expl} = \{A_i, ..., A_j, ..., A_q\}$; {conjunto de conjuntos de testes de atributos A_e correspondente às regras R_e do *ensemble* que classificam o exemplo como a mesma classe $\mathbf{h}^*(\mathbf{x})$ do *ensemble.*}
3: **if** $|\mathbf{A}_{expl}| > 1$ **then**
4: **for all** $(A_i, A_j)|i \neq j$ **do**
5: **if** $A_i \subseteq A_j$ **then**
6: $\mathbf{A}_{expl} = \mathbf{A}_{expl} - \{A_j\}$
7: **end if**
8: **end for**
9: **end if**
10: **return** $\mathbf{A}_{expl}$;

6 EXPERIMENTOS E RESULTADOS

Conforme mencionado anteriormente, em [4] são descritos experimentos realizados com o sistema ELE utilizando o conjunto de dados Nursery, da UCI [13]. Esse conjunto de dados foi extraído de um modelo de decisão hierárquica originalmente desenvolvido para classificar pedidos em berçários; foi utilizado durante a década de 80 quando existiam muitas inscrições nesses tipos de escolas em Ljubljana, Slovenia, e as inscrições negadas necessitavam de uma explicação objetiva. A decisão final depende de 3 sub-problemas: ocupação dos pais, estrutura familiar, posição social, e quadro de saúde e social da família da criança inscrita.

Na Tabela 2 são mostradas algumas característica do conjunto de dados Nursery: número de exemplos (# Ex.); número de atributos (contínuos, discretos) (# Atr.); distribuição de exemplos nas classes (Classe %) e erro majoritário do conjunto de dados. Deve ser observado que tal conjunto não possui valores desconhecidos nem exemplos duplicados ou conflitantes.

# Ex.	# Atr. (cont.,disc.)	Classe	Classe %	Erro Majoritário
12960	8 (0,8)	*not_recom*	33.33%	66.67%
		recommend	0.02%	em *not_recom*
		very_recom	2.53%	
		priority	32.92%	
		spec_prior	31.20%	

Tabela 2: Sumário das características do conjunto de dados Nursery

Na Tabela 3, são descritos os atributos do conjunto de dados.

Diversos experimentos foram realizados utilizando os algoritmos de aprendizado simbólico $\mathcal{CN}2$ [8] e $\mathcal{C}4.5$ [18] para induzir os classificadores que compõem os *ensembles* [4]. A seguir são mostrados os resultados

Atributo	Descrição
parents	Ocupação dos pais (usual, pretentious, great_pret).
has_nurs	Berçário da criança (proper, less_proper, improper, critical, very_crit).
form	Estrutura da família da criança (complete, completed, incomplete, foster).
children	Número de crianças (1, 2, 3, more).
housing	Condição residencial (convenient, less_conv, critical).
finance	Condição financeira da família (convenient, inconv).
social	Condições sociais da família (non_prob, slightly_prob, problematic).
health	Condições de saúde da família (recommended, priority, not_recom).

Tabela 3: Descrição dos atributos do conjunto de dados Nursery

de 5 (cinco) desses experimentos, os quais foram realizados variando-se o número de amostras de exemplos (retiradas do conjunto de exemplos inicial sem reposição), ou seja, variando-se o número de classificadores que compõem os *ensembles* e variando-se o algoritmo de aprendizado usado em cada amostra para induzir o classificador componente, como é mostrado na Tabela 4. Por exemplo, no primeiro experimento (*Exp 1*), 3 (três) amostras foram retiradas do conjunto inicial e foi utilizado o algoritmo $\mathcal{CN}2$ para induzir os 3 classificadores componentes, enquanto que no experimento *Exp 4*, foram retiradas 5 (cinco) amostras, sendo que $\mathcal{CN}2$ foi utilizado em 3 (três) amostras e $\mathcal{C}4.5$ em 2 (duas) amostras[3].

Experimento	# de Classificadores	Algoritmos de AM
Exp 1	3	$\mathcal{CN}2$- $\mathcal{CN}2$- $\mathcal{CN}2$
Exp 2	3	$\mathcal{C}4.5$- $\mathcal{C}4.5$- $\mathcal{C}4.5$
Exp 3	5	$\mathcal{CN}2$- $\mathcal{CN}2$- $\mathcal{CN}2$- $\mathcal{CN}2$- $\mathcal{CN}2$
Exp 4	5	$\mathcal{CN}2$- $\mathcal{CN}2$- $\mathcal{CN}2$- $\mathcal{C}4.5$- $\mathcal{C}4.5$
Exp 5	5	$\mathcal{C}4.5$- $\mathcal{C}4.5$- $\mathcal{C}4.5$- $\mathcal{C}4.5$- $\mathcal{C}4.5$

Tabela 4: Descrição dos experimentos

Na Tabela 5 são mostrados os resultados obtidos nesses 5 (cinco) experimentos. Inicialmente, para cada amostra S_l utilizada no experimento, é mostrada a taxa de erro do classificador induzido com essa amostra, e o erro padrão entre parênteses. Tal taxa de erro foi estimada com a técnica de *10-fold cross-validation* estratificado. Na seqüência, são mostrados os resultados obtidos com cada um dos três métodos de combinação de classificadores utilizados, ou seja, a taxa de erro do *ensemble* final, também estimado com a técnica de *10-fold cross-validation* estratificado. Pode ser observado na Tabela 5 que em todos os experimentos a taxa de erro dos *ensembles* de classificadores é menor que a taxa de erro dos classificadores que os compõem. Esse resultado tem 95% de confiança segundo o teste de hipóteses t. Também é interessante observar que o erro dos *ensembles* é menor, ainda quando usado o mesmo algoritmo de aprendizado (*Exp 1*, *Exp 2*, *Exp 3* e *Exp 5*), ou seja, quando a variabilidade dos classificadores induzidos é mais restrita.

Com relação aos três métodos de combinação utilizados, *i.e* UV, WMV e WMSV, não foi observada diferença significativa entre eles para este conjunto de dados. Ainda, nos experimentos *Exp 2* e *Exp 5*, nos quais todos os classificadores foram induzidos com $\mathcal{C}4.5$, foram observados resultados idênticos entre os três métodos de combinação propostos. Em todos os experimentos realizados, o algoritmo $\mathcal{C}4.5$ foi utilizado com a opção de induzir árvores de decisão, as quais foram transformadas em regras **if-then**. Os resultados na Tabela 5 mostram que, utilizando somente árvores de decisão para compor o *ensemble*, variar o método de combinação de classificadores dentre os três propostos não altera a taxa de erro dos *ensembles* construídos para este conjunto de dados, pois, como mencionado na Seção 3, as regras que compõem uma árvore de decisão são disjuntas.

Para avaliar o método de simplificação de explicação proposto neste trabalho, foram realizados experimentos com o método de simplificação de explicação nos *ensembles* construídos com os 3 (três) métodos de combinação propostos — UV, WMV e WMSV — para os 5 (cinco) experimentos — Tabela 4.

[3]O conjunto de dados utilizado para ilustrar o sistema é de médio porte. Assim, ele pode ser manipulado diretamente por ambos os algoritmos de aprendizado. A título de informação, a taxa de erro (e seu respectivo erro padrão) utilizando $\mathcal{CN}2$ e $\mathcal{C}4.5$ em todo o conjunto de dados é, respectivamente, 2.02 (0.19) e 2.92 (0.19)

Amostra	*Exp 1*	*Exp 2*	*Exp 3*	*Exp 4*	*Exp 5*
S_1	5.86	6.25	7.79	7.90	7.81
	(0.34)	(0.26)	(0.17)	(0.15)	(0.25)
S_2	5.70	6.03	8.01	7.82	7.63
	(0.26)	(0.22)	(0.19)	(0.27)	(0.26)
S_3	6.06	6.27	7.74	7.54	7.66
	(0.17)	(0.14)	(0.19)	(0.21)	(0.25)
S_4	-	-	8.19	7.82	7.75
	-	-	(0.15)	(0.30)	(0.23)
S_5	-	-	7.90	7.83	8.07
	-	-	(0.27)	(0.35)	(0.28)
UV	4.38	4.95	5.68	5.31	6.39
	(0.15)	(0.17)	(0.16)	(0.19)	(0.14)
WMV	4.31	4.95	5.45	5.19	6.39
	(0.15)	(0.17)	(0.19)	(0.14)	(0.14)
WMSV	4.17	4.95	5.42	5.33	6.39
	(0.21)	(0.17)	(0.21)	(0.17)	(0.14)

Tabela 5: Resultados obtidos com o conjunto de dados Nursery na fase de construção dos *ensembles* de classificadores.

Para avaliar a performance do algoritmo de simplificação de explicação, foi analisada a taxa de redução do número de regras entre o conjunto $\mathbf{A}_{expl}$ inicial e o conjunto $\mathbf{A}_{expl}$ final construído utilizando o Algoritmo 1. Esse conjunto final será denominado $\mathbf{A}'_{expl}$. Dado um exemplo $\mathbf{x}$, a taxa de redução no número de regras entre o conjunto inicial $\mathbf{A}_{expl}(\mathbf{x})$ e o conjunto final $\mathbf{A}'_{expl}(\mathbf{x})$ da explicação da classificação do exemplo $\mathbf{x}$, $TR(\mathbf{x})$, é definida pela Equação 1.

$$TR(\mathbf{x}) = \frac{|\mathbf{A}_{expl}(\mathbf{x})| - |\mathbf{A}'_{expl}(\mathbf{x})|}{|\mathbf{A}_{expl}(\mathbf{x})|}. \quad (1)$$

Para obter a média e o erro padrão da taxa de redução para cada *ensemble* construído, foram utilizadas as taxas de redução de cada exemplo no conjunto de dados Nursery. Na Tabela 6 são mostrados os resultados obtidos nessa fase de simplificação. Especificamente, são mostrados o número médio de regras em $\mathbf{A}_{expl}$ e $\mathbf{A}'_{expl}$ e o erro padrão correspondente, bem como a média da taxa de redução nesse conjunto de dados.

Como pode ser observado na Tabela 6, os resultados obtidos são bastante significativos, tendo em vista que há uma redução no número de regras de, na média, mais de 50% em todos os experimentos realizados, ou seja, o número de regras de explicação para cada exemplo foi, na média, reduzido pela metade, sendo que em muitos casos, é observada uma redução muito maior. Verificando as regras que cobrem cada exemplo $\mathbf{x}$, *i.e* $\mathbf{A}_{expl}(\mathbf{x})$ e $\mathbf{A}'_{expl}(\mathbf{x})$, foi observado que realmente foram removidas regras mais específicas e não somente regras idênticas[4]. Em relação às classes com pequena distribuição de exemplo, para a classe *very_recom* (2.53% dos exemplos — Tabela 2), os exemplos dessa classe foram corretamente classificados pelo *ensemble*. Já em relação à classe *recommend* (0.02% dos exemplos — Tabela 2), como o número de exemplos é excessivamente pequeno, devido ao método de amostragem utilizado (sem reposição), o *ensemble* não conseguiu classificar exemplos dessa classe.

Pode ser também observado na Tabela 6 que nos experimentos *Exp 2* e *Exp 5*, nos quais todos os classificadores que constituem o *ensemble* são induzidos utilizando $\mathcal{C}4.5$, os resultados para os três métodos de combinação são iguais. Esse resultado era esperado já que existe no máximo somente uma regra que cobre o exemplo, pois foi utilizada a opção de indução de árvores de decisão com esse algoritmo de aprendizado ($\mathcal{C}4.5$). Isso também pode ser observado nos experimentos realizados na fase de construção dos *ensembles* (*Exp 2* e *Exp 5* — Tabela 5).

Com o objetivo de ilustrar a simplificação de explicação, nas Tabelas 7 e 8 são mostrados os resultados de simplificação de explicação para 2 (dois) exemplos diferentes no experimento *Exp 1* utilizando o método de combinação UV. Nessas tabelas, na primeira linha é mostrado o exemplo a ser classificado e a classe verdadeira do exemplo. Na segunda linha encontra-se a classificação fornecida pelo *ensemble*. Após, para

[4]Na realidade, na implementação do Algoritmo 1, $\mathbf{A}_{expl}$ é construído utilizando os corpos de todas as regras correspondentes, ou seja, das regras pertencentes ao *ensemble* que cobrem o exemplo $\mathbf{x}$ cuja cabeça prediz a classe $\mathbf{h}^*(\mathbf{x})$.

Método de Combinação		*Exp 1*	*Exp 2*	*Exp 3*	*Exp 4*	*Exp 5*
$\|\mathbf{A}_{expl}\|$	UV	4.04	2.92	6.52	5,77	4,78
		(2.20)	(0.26)	(3.22)	(2,06)	(0,55)
	WMV	4.04	2.92	6.51	5,77	4,78
		(2.20)	(0.26)	(3.24)	(2,08)	(0,55)
	WMSV	4.04	2.92	6.51	5,77	4,78
		(2.20)	(0.26)	(3.25)	(2,07)	(0,55)
$\|\mathbf{A}'_{expl}\|$	UV	1.70	1.17	2.01	1,80	1,16
		(1.11)	(0.38)	(1.41)	(1,16)	(0,40)
	WMV	1.69	1.17	2.00	1,80	1,16
		(1.12)	(0.38)	(1.42)	(1,16)	(0,40)
	WMSV	1.69	1.17	2.00	1,80	1,16
		(1.12)	(0.38)	(1.42)	(1,16)	(0,40)
$TR\%$	UV	58.19	59.43	68.88	68,88	75,02
		(16.19)	(14.22)	(17.46)	(16,25)	(10,61)
	WMV	58.18	59.43	68.76	68,79	75,02
		(16.22)	(14.22)	(17.75)	(16,49)	(10,61)
	WMSV	58.18	59.43	68.77	68,92	75,02
		(16.21)	(14.22)	(17.39)	(16,15)	(10,61)

Tabela 6: Resultados obtidos com o conjunto de dados Nursery na fase de simplificação de explicação.

cada um dos classificadores que compõem o *ensemble*, são mostrados os corpos das regras que cobrem o exemplo com a classe determinada pelo *ensemble*. Finalmente, é mostrado o conjunto dos corpos das regras a ser mostrado ao usuário para explicar a classificação do *ensemble* para esse exemplo. Deve ser observado que corpos de regras iguais são enumerados com o mesmo valor numérico (índice) e, caso uma regra for especialização de outra regra, o corpo da regra é enumerado com o índice da regra mais geral seguido do símbolo *. Por exemplo, na Tabela 7, os dois corpos de regras do classificador 2 e o corpo da regra do classificador 3 identificados por 4* são especializações do corpo da regra 4.

Nas Tabelas 7 e 8, pode-se observar que regras especializadas de outras são induzidas, e que o algoritmo para simplificação de explicação é bastante útil para usuários do sistema ELE, já que diminui bastante a quantidade de regras que explicam ao usuário/especialista a decisão tomada pelo *ensemble*.

É interessante notar nos resultados descritos nessas tabelas que, ainda que foi utilizado o mesmo algoritmo de AM ($\mathcal{CN}2$) no experimento *Exp 1*, de onde foram extraídos esses resultados, são induzidas algumas regras distintas nos diferentes classificadores. Isso se deve ao fato do $\mathcal{CN}2$ induzir regras não ordenadas e não disjuntas, ao contrário das regras que representam a árvore de decisão induzidas pelo $\mathcal{C}4.5$, que também são não ordenadas mas são disjuntas.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foram propostos três métodos de construção de *ensembles* de classificadores simbólicos e um método para simplificar a explicação fornecida pelos *ensembles* simbólicos construídos com os métodos aqui propostos. Também, foram descritos os sistemas computacionais implementados para avaliar esses métodos.

Foi conduzida uma série de experimentos usando um conjunto de dados da UCI com tamanho limitado. Apesar de para este conjunto de dados não haver diferença significativa entre os três métodos de combinação propostos, a taxa de erro estimada dos *ensembles* construídos é menor que a taxa de erro de cada um dos classificadores que o compõem, com 95% de confiança segundo o teste de hipóteses t. Assim, é interessante observar que a técnica, a qual contempla o uso de grandes bases de dados, também pode ser utilizada para extrair conhecimento de conjuntos de dados de tamanho menor. Ainda, em todos os experimentos realizados com o método de simplificação de explicação, houve uma taxa significativa de redução de regras. Esses resultados são muito encorajadores.

Como trabalhos futuros, serão conduzidos mais experimentos utilizando outros conjuntos de dados, tanto de grande quanto de pequeno porte, variando o método de amostragem, o número L de classificadores induzidos e os algoritmos de aprendizado simbólicos utilizados para induzir os classificadores do *ensemble*.

	Exemplo $\mathbf{x}_1$: (usual,proper,complete,1,convenient,convenient,problematic,priority) - Classe: priority
	Método de Combinação: UV - Experimento: *Exp 1* - Classificação: priority
	Corpo das regras com classe igual a priority que cobrem $\mathbf{x}_1$ (que compõem A_{expl}):
	Classificador 1:
1	parents = usual AND has_nurs = proper AND health = priority
2	has_nurs = proper AND housing = convenient AND finance = convenient AND health = priority
3	form = complete AND children = 1 AND housing = convenient AND finance = convenient AND health = priority
4	has_nurs = proper AND children = 1 AND health = priority
	Classificador 2:
1	parents = usual AND has_nurs = proper AND health = priority
3	form = complete AND children = 1 AND housing = convenient AND finance = convenient AND health = priority
2	has_nurs = proper AND housing = convenient AND finance = convenient AND health = priority
4*	has_nurs = proper AND form = complete AND children = 1 AND health = priority
4*	has_nurs = proper AND children = 1 AND housing = convenient AND social = problematic AND health = priority
	Classificador 3:
1	parents = usual AND has_nurs = proper AND health = priority
2	has_nurs = proper AND housing = convenient AND finance = convenient AND health = priority
4*	has_nurs = proper AND form = complete AND children = 1 AND health = priority
3	form = complete AND children = 1 AND housing = convenient AND finance = convenient AND health = priority
	Corpo das regras para explicação ao usuário (que compõem A'_{expl}):
1	parents = usual AND has_nurs = proper AND health = priority
2	has_nurs = proper AND housing = convenient AND finance = convenient AND health = priority
3	form = complete AND children = 1 AND housing = convenient AND finance = convenient AND health = priority
4	has_nurs = proper AND children = 1 AND health = priority

Tabela 7: Resultados obtidos com o conjunto de dados Nursery na fase de simplificação de explicação — Exemplo $\mathbf{x}_1$.

	Exemplo $\mathbf{x}_2$: (usual,proper,complete,1,less_conv,inconv,problematic,priority) - Classe: priority
	Método de Combinação: UV - Experimento: *Exp 1* - Classificação: priority
	Corpo das regras com classe igual a priority que cobrem $\mathbf{x}_2$ (que compõem A_{expl}:)
	Classificador 1:
1	parents = usual AND has_nurs = proper AND health = priority
2	has_nurs = proper AND children = 1 AND health = priority
	Classificador 2:
1	parents = usual AND has_nurs = proper AND health = priority
2*	has_nurs = proper AND children = 1 AND housing = less_conv AND health = priority
2*	has_nurs = proper AND form = complete AND children = 1 AND health = priority
	Classificador 3:
1	parents = usual AND has_nurs = proper AND health = priority
2*	has_nurs = proper AND form = complete AND children = 1 AND health = priority
	Corpo das regras para explicação ao usuário (que compõem A'_{expl}):
1	parents = usual AND has_nurs = proper AND health = priority
2	has_nurs = proper AND children = 1 AND health = priority

Tabela 8: Resultados obtidos com o conjunto de dados Nursery na fase de simplificação de explicação — Exemplo $\mathbf{x}_2$.

Com relação à explicação, atualmente o sistema mostra ao usuário/especialista o conjunto de regras diferentes e mais gerais que cobrem o exemplo com a classe determinada pelo *ensemble*. Entretanto, após encontrado esse conjunto de regras, é possível ordenar essas regras utilizando diferentes medidas de avaliação de regras, tais como precisão, sensibilidade, especificidade, suporte, novidade e outras [14, 5]. Dessa forma, o usuário poderia especificar uma medida de avaliação a ser utilizada e essas regras seriam mostradas em ordem de importância segundo essa medida. Essa extensão será implementada futuramente.

AGRADECIMENTOS

Os autores agradecem as valiosas sugestões dos revisores.

Referências

[1] G. E. A. P. A. Batista and M. C. Monard. Descrição da arquitetura e do projeto do ambiente computacional Discover Learning Environment - DLE. Technical Report 187, ICMC/USP, 2003. `ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_187.PDF`.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36(1/2):105–139, 1999.

[3] F. C. Bernardini. Combinação de classificadores utilizando medidas de avaliação de regras e algoritmos evolutivos, 2003. Exame de Qualificação de Doutorado, ICMC/USP.

[4] F. C. Bernardini and M. C. Monard. Methods for constructing symbolic ensembles from symbolic classifiers. In *Fifth Congress of Logic Applied to Technology – LAPTEC 2005 – Advances in Intelligent Systems and Robotics*, Japão, 2005. (in print).

[5] F. C. Bernardini, M. C. Monard, H. D. Lee, and S. Esteves. Um algoritmo para selecionar regras de conhecimento utilizando medidas de avaliação de regras. In *IV Workshop de Inteligência Artificial – Jornadas Chilenas de Computacion*, pages 1–7, Chillan, Chile, 2003.

[6] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] L. Breiman. Arcing classifiers. *The Annals Of Statistics*, 26(3):801–849, 1998.

[8] P. Clark and T. Niblett. The $\mathcal{CN}2$ induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[9] T. G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18:97–136, 1997. `http://www.cs.orst.edu/~tgd/`.

[10] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40:139–157, 2000. `http://www.cs.orst.edu/~tgd/`.

[11] T. G. Dietterich. *Ensemble* methods in machine learning. In *First International Workshop on Multiple Classifier Systems. Lecture Notes in Computer Science*, volume 1857, pages 1–15, New York, 2000. Springer Verlag.

[12] Y. Freund and R.E.. Schapire. A decision-theoretic generalization of on-line learninng and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[13] S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[14] N. Lavrac, P. Flach, and B. Zupan. Rule evaluation measures: a unifying view. In *Proc. 9th International Workshop on Inductive Logic Programming. Lecture Notes in Artificial Intelligence*, volume 1634, pages 74–185, New York, 1999. Springer Verlag.

[15] K. T. Leung and D. S. Parker. Empirical comparisons of various voting methods in bagging. In *KDD '03: Proc. 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 595–600. ACM Press, 2003.

[16] R. C. Prati, J. A. Baranauskas, and M. C. Monard. Extração de informações padronizadas para a avaliação de regras induzidas por algoritmos de aprendizado de máquina simbólico. Technical Report 145, ICMC/USP, 2001. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_145.ps.zip`.

[17] R. C. Prati, M. R. Geromini, and M. C. Monard. A framework to integrate data mining components. In *XXIX Conferencia Latinoamericana de Informatica — CLEI2003*, pages 1–11, Bolivia, 2003.

[18] J. R. Quinlan. *C*4.5 *Programs for Machine Learning.* Morgan Kaufmann Publishers, Inc., 1988.

[19] L. Todorovski and S. Dzeroski. Combining classifiers with meta decision trees. *Machine Learning*, 50(3):223–249, 2003.

[20] L. Wall, T. Christiansen, and R. Schwartz. *Programming in PERL.* Morgan Kaufmann Publishers, Inc., 1999.

Uma Proposta de Novo Método de Seleção Roleta para os Algoritmos Genéticos Baseado no Paradigma do Dilema do Prisioneiro

Otávio Noura Teixeira
Centro Universitário do Pará, Área de Ciências Exatas e Tecnologia,
Belém, Brasil, 66.023-230
onoura@gmail.com

Roberto Célio Limão de Oliveira
Universidade Federal do Pará, Departamento de Engenharia Elétrica e Computação,
Belém, Brasil, 66.075-900
limao@ufpa.br

Abstract

This article presents a proposal for a new roulette selection method for Genetic Algorithms (GA) based on the Prisoner's Dilemma Paradigm – the most classic two person non-zero-sum non-cooperative game in Game Theory. In this way, the Roulette Prisoner's Dilemma (RPD) was developed. It permits the race of available resources by individuals of the population. Therefore, they have the possibility of alter their adaptability, then influencing on the generation of offspring. Besides that, some simulation results are presented and they are compared with other selection methods: Roulette and Roulette Hawk-Dove (RHD).

Keywords: Genetic Algorithms, Game Theory, Prisoner's Dilemma, Selection Method, Traveling Salesman Problem.

Resumo

Este artigo apresenta uma proposta de um novo método de seleção roleta para os Algoritmos Genéticos (AG) baseado no Paradigma do Dilema do Prisioneiro (DP) – o mais clássico dos jogos de duas pessoas soma não-zero não-cooperativos da Teoria dos Jogos –, originando assim o método de seleção Dilema do Prisioneiro Roleta (DPR), que permite aos indivíduos da população disputar por recursos disponibilizados pelo jogo. Dessa forma, eles têm a possibilidade de alterar a sua adaptabilidade, o que conseqüentemente influência na geração de descendência. Além disso, os resultados de algumas simulações são apresentados e comparados com os seguintes métodos de seleção: Roleta e Hawk-Dove Roleta (HDR).

Palavras Chave: Algoritmos Genéticos, Teoria dos Jogos, Dilema do Prisioneiro, Método de Seleção, Problema do Caixeiro Viajante.

1. INTRODUÇÃO

A Computação Evolucionária é uma área que busca inspiração no processo de otimização utilizado pela Natureza, a Evolução. Baseando-se sempre no princípio de que, se de algumas cadeias de moléculas orgânicas a Natureza foi capaz de criar seres tão complexos capazes de perceberem e controlarem a própria existência, simular este processo e adaptá-lo a solução de problemas e a otimização de soluções trará resultados bastante satisfatórios [7].

Na área de Computação Evolucionária, destacam-se os Algoritmos Genéticos (AG), que baseados nos conceitos de genética e da Seleção Natural de Charles Darwin, modelam os operadores biológicos de seleção, recombinação e mutação, para otimizar funções, resolver problemas e encontrar soluções com maior "adaptabilidade" (*fitness*), ou seja, melhor desempenho [11].

Os Algoritmos Genéticos, apesar de possuírem alta aplicabilidade em muitas áreas do conhecimento e com muitos casos de sucesso, ainda possuem diversos aspectos a serem estudados e otimizados. Dentre esses aspectos,

temos aqueles que, principalmente, expõem os AG a um alto grau de aleatoriedade, o que vai de encontro aos princípios da própria Evolução.

Neste sentido, muitos pesquisadores procuram agregar novas variáveis biológicas (biologicamente plausíveis) à estrutura tradicional de AG, buscando reduzir o seu grau de aleatoriedade, atingindo assim, melhores resultados com um menor tempo de execução [14].

Além disso, Melanie Mitchell em seu livro "An Introduction to Genetic Algorithms" afirma que no contexto dos Algoritmos Genéticos não há a noção de *fenótipo*, apesar de recentemente alguns trabalhos apresentarem alguns experimentos onde ambos os aspectos são considerados: o genótipo e o fenótipo [14].

Dessa forma, os primeiros métodos de seleção, no intuito de possibilitar a alteração da adaptabilidade dos indivíduos antes da fase de reprodução, foram apresentados em [12], ao propor três novos métodos baseados no jogo Hawk-Dove – que é um jogo evolucionário –, o Hawk-Dove Roleta (HDR) e suas variações. Na seqüência, o método Hawk-Dove Torneio (HDT) apresentado em [4] e [5], foi proposto.

Como forma de confrontar os jogos evolucionários e os jogos sociais é que surgiu a motivação para o desenvolvimento do método de seleção apresentado em [16]. E assim, este artigo apresenta uma nova operação de seleção para os Algoritmos Genéticos baseada no jogo Dilema do Prisioneiro, o Dilema do Prisioneiro Roleta (DPR), e alguns resultados preliminares em relação aos métodos da Roleta Simples e Hawk-Dove Roleta (HDR) são apresentados ao final.

2. ALGORITMOS GENÉTICOS

O enunciado "a biologia é um processo de otimização" é a filosofia de suporte dos métodos da Computação Evolucionária, tendo os Algoritmos Genéticos (AG) como um dos seus principais paradigmas.

Os Algoritmos Genéticos são os algoritmos evolucionários mais utilizados da Computação Evolucionária, e estão baseados nos princípios da Teoria da Evolução Biológica dos Seres Vivos, e são utilizados na resolução de problemas computacionais. Tais problemas são freqüentemente os que requerem uma busca através de um amplo espaço de soluções candidatas [10].

Inspirados na teoria de Darwin apresentada através de seu livro "Origem das Espécies", onde uma população natural durante várias gerações, desenvolve-se segundo os princípios da Seleção Natural e da "Sobrevivência do Mais Apto", os AG são capazes de desenvolver soluções para problemas do mundo real, tais como problemas de busca e otimização [1].

Esta técnica de resolução de problemas foi desenvolvida por John Holland e apresentada em seu livro "Adaptation in Natural and Artificial Systems" de 1975, onde apresentou os Algoritmos Genéticos como uma abstração da evolução biológica, proporcionando assim toda a sua base teórica.

Em [10], os AG são definidos como um método de se movimentar de uma população a outra, utilizando um tipo de "Seleção Natural" em conjunto com outros operadores genéticos, tais como: mutação, onde um gene é substituído por um alelo de forma aleatória; e, recombinação, em que os cromossomos escolhidos trocam pedaços gerando novas soluções. Este processo é repetido até que a solução ou conjunto de soluções satisfatórias seja obtido.

De uma forma geral, os AG são algoritmos de busca bastante eficientes que implementam de uma maneira primitiva alguns dos processos da evolução natural. São ideais para otimizações de funções e classificação de padrões, pois com sua estrutura conseguem realizar uma busca eficiente em um espaço muito grande de soluções [7].

Atualmente, apesar do conceito dos AG ter absorvido outras áreas e, além disso, ter se aproximado mais ainda da simulação biológica, a estrutura proposta por Holland ainda é válida e defendida por muitos autores, sendo chamada AG Simples [10].

Esta forma consiste em: gerar uma população de soluções (cromossomos) modeladas de acordo com o problema estudado; avaliar a adaptabilidade (*fitness*) de cada um, que pode ser compreendida como a probabilidade de uma solução gerar novos descendentes com seu material genético; e, aplicar os operadores de seleção, recombinação e mutação sobre as soluções, repetindo o processo até que seja encontrado um resultado satisfatório ou um número máximo de gerações seja atingido [11].

De acordo com o que está especificado em [12], a operação de seleção irá selecionar cromossomos da população, utilizando na maioria das vezes, o valor de *fitness*, possibilitando aos indivíduos mais aptos gerar um maior número de descendentes, ou seja, passar pelo operador de recombinação mais vezes. Assim, temos uma relação direta com o Processo de Seleção Natural, apresentado por Charles Darwin em sua obra "Origem das Espécies" [3].

Para a implementação deste operador, diversos métodos foram propostos, onde se destacam o método da Roleta ou Seleção Proporcional ao *fitness*, caracterizado pela tradição e facilidade de implementação; e o método do Torneio, mais recente e tido como um dos métodos mais eficientes entre os vários métodos de seleção, tanto computacionalmente – O(n) –, quanto na exploração da superfície de busca do AG [15].

O Operador de Recombinação ou *Crossover* é a principal característica que distingue os AG de outros métodos de busca, e é considerado o mais importante entre os operadores genéticos. Além disso, é a primeira maneira

que um algoritmo genético usa para explorar a superfície de busca dentro do material genético (alelos) que já se encontra nos indivíduos da população, levando o AG à convergência de resultados [11].

A recombinação, de um modo geral, consiste na troca de pedaços (partes) de cromossomos entre os indivíduos selecionados como pais, para a criação de novos indivíduos melhores adaptados, podendo apresentar diversos métodos como os tradicionais: um ponto de corte, dois pontos de corte e parcialmente mapeado (PMX) [9].

O Operador de Mutação, segundo [11], é a segunda maneira que um AG utiliza para explorar a superfície de busca do problema. Ele pode introduzir um alelo que não estava na população e evitar a sua rápida convergência, "escapando" de mínimos locais.

A operação de mutação funciona fazendo pequenas alterações dos genes da população, substituindo o conteúdo de um gene por um elemento do alfabeto de alelos de forma aleatória e com uma pequena probabilidade para cada gene. [7].

3. TEORIA DOS JOGOS E O DILEMA DO PRISIONEIRO

Humanos, animais e todos os seres vivos de uma forma geral participam de jogos, que vão de uma simples disputa de xadrez às manobras políticas internacionais, das disputas para conquistar o sexo oposto às disputas por um determinado mercado, que são travadas por duas grandes empresas.

A Teoria dos Jogos pode ser definida como a descrição formal do conflito de interesses entre participantes, em que todos fazem escolhas buscando a melhor estratégia, onde o conjunto de escolhas de todos os participantes determinará os resultados obtidos por cada um deles [13].

O estudo formal, do que hoje se chama Teoria dos Jogos, é um campo relativamente recente e tem mostrado alta aplicabilidade prática para diversas áreas, caminhando de uma ferramenta de análise matemática para guiar apostadores, para a revolução dos paradigmas de áreas como a economia e a biologia evolucionária [13].

Em [2] é afirmado que em 1950, Melvin Dresher e Merrill Flood, dois cientistas da RAND Corporation apresentaram um jogo peculiar, e que mais tarde tornou-se o conhecido Dilema do Prisioneiro. De acordo com alguns autores, este é o jogo de duas pessoas não-cooperativo de soma não-zero mais famoso e, também, o mais conhecido da Teoria dos Jogos.

A seguir na tabela 1 é possível verificar a tabela de pagamentos para este jogo.

J_1 / J_2	**Cooperar**	**Trair**
Cooperar	(R,R)	(S,T)
Trair	(T,S)	(P,P)

Tabela 1: Tabela de pagamentos do dilema do prisioneiro

Cada jogador pode expressar dois comportamentos diferentes, que são: *Cooperar* (Cooperate ou C) e *Trair* (Defect ou D). E como forma de constituir um Dilema do Prisioneiro, os valores, representados por *T*, *R*, *P* e *S* nas células da tabela, devem obedecer as seguintes relações:

a. $T > R > P > S$
b. $R > (T+S)/2$
c. $(T+S)/2 > 2$

Para cada par de comportamentos possível, o pagamento (R, T, P e S) de linha-coluna está listado na célula apropriada da tabela, onde cada valor significa:

R é a *recompensa* (Reward) que cada jogador recebe se ambos os jogadores cooperarem;
P é a *punição* (Punishment) que cada um dos jogadores recebe quando ambos traem;
T é a *tentação* (Temptation) que cada jogador tem, caso traia sozinho;
S é o pagamento do *otário* (Sucker) que coopera sozinho.

Aqui é assumido que o jogo é simétrico, ou seja, que os valores de recompensa, tentação, punição, e de ser otário são os mesmos para cada um dos jogadores, e esses valores apresentam um significado em relação a sua ordem, isto é, indica quando um valor de pagamento é melhor do que outro, mas não informa o quanto é melhor.

Suponhamos os seguintes jogadores A e B, representados respectivamente por J_A e J_B. Desta forma, caso J_B coopere, então J_A recebe R por cooperar e T por trair, o que é lhe é mais vantajoso. Caso contrário, supondo que J_B traia, então J_A recebe S por cooperar e P por trair, e a traição novamente é a melhor opção para este jogador.

O comportamento "trair" para J_A é chamado de *estritamente dominante* em relação ao comportamento "cooperar", independentemente do que o J_B faça, pois ele resulta em um pagamento melhor do que cooperar. Por simetria, o mesmo ocorre com o J_B.

Desta forma, dois jogadores ditos "racionais" trairiam e receberiam um pagamento igual a P, enquanto que outros dois jogadores ditos "irracionais" cooperariam e receberiam um pagamento igual a R, que é maior que P.

Normalmente, a Teoria dos Jogos assume que a racionalidade é de *conhecimento comum*, onde cada jogador é racional e também sabe que o outro é racional, e sabe que o outro sabe que ele é racional. Cada jogador também sabe quais são os outros valores de pagamento, mas a partir do momento em que "trair" domina "cooperar", e isso para os dois jogadores, o argumento para o dilema requer simplesmente que cada jogador saiba sobre os seus próprios pagamentos.

É importante notar que o par de comportamentos "trair / trair" apresenta o único e forte *Equilíbrio de Nash* no jogo, ou seja, é a única saída onde cada jogador só poderá obter valores piores caso troquem unilateralmente de comportamento.

4. DILEMA DO PRISIONEIRO ROLETA (DPR)

Este método permite que os indivíduos alterem o seu valor de adaptabilidade por determinado período de tempo, onde é permitido lutar pela melhoria de suas aptidões, disputando o Jogo *Dilema do Prisioneiro* contra outros indivíduos.

Isto ocorre antes da formação da roleta, que passará a levar em consideração não apenas o valor genotípico de adaptabilidade, mas também o acréscimo ou decréscimo dado às disputas realizadas entre os indivíduos, ou seja, a nível fenotípico.

Para que isto ocorra, antes da aplicação do operador de seleção, os indivíduos participam de x disputas. Estas disputas são realizadas escolhendo dois indivíduos da população de forma aleatória, comparando as estratégias utilizadas por cada um deles e, assim, alterando suas adaptabilidades conforme a tabela de pagamento do Jogo *Dilema do Prisioneiro*, apresentada na tabela 1.

Após as disputas, é gerada uma roleta proporcional à nova adaptabilidade dos indivíduos, para que seja realizada a operação de seleção através do uso de um ponteiro aleatório, da mesma forma como ocorre no operador de seleção roleta.

A seguir é apresentado o algoritmo para este método de seleção.

1. Gerar população inicial;
2. Avaliar cada indivíduo da população inicial;
3. Repetir até m gerações:
   - 3.1. Repetir até x disputas:
     - 3.1.1. Selecionar dois indivíduos aleatoriamente;
     - 3.1.2. Obter o comportamento de cada um dos indivíduos;
     - 3.1.3. Alterar a adaptabilidade dos indivíduos conforme o comportamento adotado por cada um, e a tabela de pagamentos do jogo;
   - 3.2. Montar a roleta;
   - 3.3. Repetir até o número de descendentes ser igual à quantidade desejada (% de cruzamento)
     - 3.3.1. Selecionar dois indivíduos utilizando a roleta;
     - 3.3.2. Realizar a operação de cruzamento nos indivíduos selecionados no passo anterior;
     - 3.3.3. Realizar a operação de mutação nos descendentes gerados no passo anterior;
     - 3.3.4. Avaliar os descendentes;
4. Substituir os indivíduos da população pelos novos indivíduos gerados na etapa de reprodução;
5. O melhor indivíduo da população é a solução do problema.

A seguir é realizada uma exemplificação do algoritmo para o método de seleção Dilema do Prisioneiro Roleta.

5. SIMULAÇÃO DO ALGORITMO DPR

Como forma de exemplificar o funcionamento desse método e demonstrar a diferença entre os jogos Hawk-Dove e Dilema do Prisioneiro, na seqüência, na tabela 2, é apresenta a tabela de pagamentos que utilizaremos para exemplificar o funcionamento do jogo Dilema do Prisioneiro. A seqüência de pares de indivíduos participantes das disputas é a mesma utilizada para a exemplificação do método *HDR*.

J_1 / J_2	Cooperar	Trair
Cooperar	(25,25)	(10,30)
Trair	(30,10)	(15,15)

Tabela 2: Tabela de pagamentos utilizada na exemplificação

De posse da tabela de pagamentos para o jogo, ainda é necessário realizar a geração da população inicial, a sua avaliação e as estratégias de comportamento dos indivíduos dessa população.

Supondo então que essas etapas já tenham sido realizadas, na figura a seguir temos a população inicial que, para efeitos comparativos, é a mesma utilizada na simulação da execução do algoritmo para o método *HDR*.

Cromossomo	Aptidão	Estratégia			
1	100	*Cooperar*	2	60	*Trair*
3	130	*TFT*	4	110	*Aleatório*

Figura 1: População antes da aplicação do método

Como a primeira fase do jogo é selecionar dois indivíduos, eles são selecionados de forma aleatória, e com probabilidades iguais 0,25, exatamente da mesma maneira realizada para o jogo Hawk-Dove.

Suponhamos que para a **primeira partida** do jogo são selecionados os jogadores **2** e **3**, com as respectivas estratégias de comportamento, *ALL-D* (sempre trair) e *TFT*. Este verifica se já disputou o jogo anteriormente com o indivíduo 2, e assim, o indivíduo 3 adota o comportamento *Cooperar*, devido esta ser a primeira partida entre eles.

Com os comportamentos escolhidos é configurada a situação *Trair* vs *Cooperar* e, conseqüentemente, o indivíduo 2 tem um ganho de trinta (30) unidades em sua adaptabilidade, enquanto que o indivíduo 3 obtém um ganho de apenas dez (10) unidades. E assim, a população passa a seguinte configuração.

Cromossomo	Aptidão	Estratégia			
1	100	*Cooperar*	2	90	*Trair*
3	140	*TFT*	4	110	*Aleatório*

Figura 2: População após a primeira partida

Na **segunda disputa**, os indivíduos **2** e **4** são selecionados também aleatoriamente e apresentam as estratégias *ALL-D* e *Aleatório*, respectivamente. O indivíduo 4 escolhe de forma aleatória o comportamento *Trair*.

Desta forma, nesta segunda partida tem-se a situação *Trair* vs *Trair*, e conseqüentemente os dois indivíduos terão as suas aptidões aumentas em quinze (15) unidades. E assim, a população composta pelos quatro indivíduos passa a se apresentar da seguinte maneira, conforme está apresentado na figura 3.

Cromossomo	Aptidão	Estratégia			
1	100	*Cooperar*	2	105	*Trair*
3	140	*TFT*	4	125	*Aleatório*

Figura 3: População após a segunda partida

Na **terceira partida** são selecionados os indivíduos **1** e **3**, que são portadores das estratégias *ALL-C* e *TFT*, e por esse motivo o indivíduo 3 verifica primeiramente se já participou de alguma partida com o indivíduo 1. Após constatar que ainda não disputou nenhuma partida com o indivíduo 1, ele adota o comportamento *Cooperar*.

Dessa forma, tem-se a situação *Cooperar* vs *Cooperar*, e conseqüentemente, cada um dos indivíduos tem a sua adaptabilidade aumentada em vinte e cinco (25) unidades, conforme pode ser visto na figura 4 a seguir.

Cromossomo	Aptidão	Estratégia			
1	125	*Cooperar*	2	105	*Trair*
3	165	*TFT*	4	125	*Aleatório*

Figura 4: População após a terceira partida

Para a **quarta partida**, que é a última, são selecionados novamente os indivíduos **2** e **3**, que participaram da primeira partida. Portanto, o indivíduo 2 tem a estratégia *ALL-D*, e o indivíduo 3, pelo fato de ter a estratégia *TFT*, verifica primeiro se já participou de alguma disputa com o indivíduo 2. Ao constatar que sim, o indivíduo 3 adota o comportamento adotado pelo indivíduo 2 na partida anterior, que é o comportamento *Trair*.

Desta maneira, a situação *Trair* vs *Trair* é estabelecida, onde cada um dos indivíduos tem a sua adaptabilidade aumentada em quinze (15) unidades. E assim, na figura a seguir, temos a nova configuração da população, após a quarta partida.

Cromossomo	Aptidão	Estratégia			
1	125	*Cooperar*	2	120	*Trair*
3	180	*TFT*	4	125	*Aleatório*

Figura 5: População após a quarta partida

Após as quatro partidas, a roleta é montada para que a operação de seleção ocorra, através da escolha de indivíduos que se submeterão à operação de cruzamento, gerando assim os novos indivíduos da população. Conseqüentemente, os processos restantes do algoritmo são executados, exatamente da mesma forma que no método *HDR*.

Com a realização das quatro partidas do jogo Dilema do Prisioneiro houve uma substancial alteração da distribuição de probabilidade dos indivíduos serem selecionados a partir de suas fatias na roleta.

O indivíduo 1 obteve uma perda de 0,023, equivalente a 2% do total; o indivíduo 2 obteve um ganho de 0,068, equivalente a 7%; o indivíduo 3 obteve um ganho insignificante de 0,002 e manteve-se com a mesma porção da roleta; e, o indivíduo 4 obteve uma perda de 0,057, equivalente a 6%. Isto é: os ganhos obtidos pelos indivíduos 2 e 3 são equivalentes aos valores que os indivíduos 1 e 4 perderam.

Na figura 6 é possível visualizar a constituição da roleta antes e depois da realização das quatro disputas.

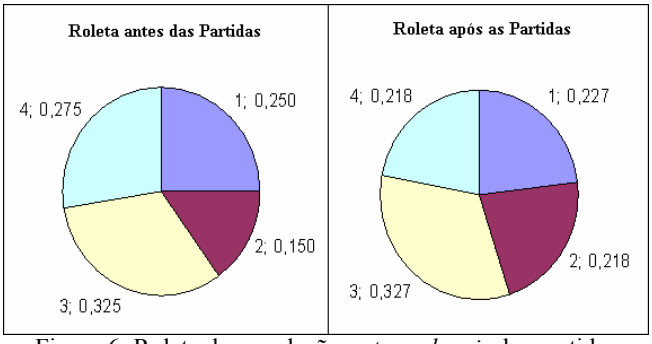


Figura 6: Roleta da população *antes* e *depois* das partidas

De acordo com a tabela de pagamentos apresentada na tabela 2, e ao analisar o jogo Dilema do Prisioneiro com apenas uma única iteração entre os indivíduos ao participarem das disputas, podemos afirmar que a tendência desses indivíduos é agir baseados no comportamento *Trair*. O ponto de equilíbrio desse jogo está na relação *Trair* vs *Trair*, que constitui o único Equilíbrio de Nash presente.

Sendo assim, podemos compreender o porquê do ganho de 0,068 obtido pelo indivíduo 2 durante a etapa de interação social, pois conseguiu dobrar o seu valor de adaptabilidade em decorrência das características das disputas e das estratégias de comportamento dos outros indivíduos.

6. AVALIAÇÃO DO MÉTODO

Para a avaliação do método proposto foi implementada uma plataforma de testes, de forma que a população inicial, a taxa de cruzamento e a taxa de mutação mantiveram iguais para todos os testes.

O problema utilizado para a avaliação do método foi o Problema do Caixeiro Viajante Simétrico (PCVS), aplicado as 26 capitais do Brasil que estão interligadas por rodovias [6].

Sendo assim, foram executados 10 testes para os seguintes métodos de seleção: dois para Roleta Simples; quatro para o Hawk-Dove Roleta (HDR), apresentado em [12]; e, quatro para o Dilema do Prisioneiro Roleta (DPR). Os resultados estão apresentados na tabela 3, utilizando para validação e comparação os seguintes parâmetros fixos:

- População inicial;
- Número de indivíduos: fixados em 100 para todas as gerações;
- Número de gerações: fixados em 5.000 e 10.000;
- Percentual de cruzamentos numa população: 85%;
- Percentual de mutação numa população: 4% ;
- Número de partidas (para os métodos que envolvem jogo): 100 por geração;
- Relação das estratégias de comportamento da população inicial: $\frac{3}{9} ALL - D : \frac{3}{9} ALL - C : \frac{2}{9} TFT : \frac{1}{9} Aleatório$

Teste	Métodos: {Geração, Partidas, Valores de Pagamento}	Melhor Indivíduo		Média (Km)
		Distância (Km)	Geração	
01	Roleta Simples: {5.000}	24.803	999	27.287,1
02	Roleta Simples: {10.000}	24.138	2.277	25.545,6
03	HDR: {5000, 100, [C,V]=[20,30]}	26.021	674	27.268,1
04	HDR: {10000, 100, [C,V]=[20,30]}	20.093	1.636	22.160,6
05	HDR: {5000, 100, [C,V]=[30,20]}	24.629	584	25.899,3
06	HDR: {10000, 100, [C,V]=[30,20]}	27.051	555	25.899,3
07	DPR: {5000, 100, [T, R, P, S]=[30, 25, 15, 10]}	23.284	761	25.917,4
08	DPR: {10000, 100, [T, R, P ,S]=[30, 25, 15, 10]}	28.605	817	30.521,3
09	DPR: {5000, 1000, [T, R, P ,S]=[30, 25, 15, 10]}	25.433	3.102	29.132,7
10	DPR: {10000, 1000, [T, R, P, S]=[30, 25, 15, 10]}	22.568	474	24.622,0

Tabela 3: Avaliação dos resultados de forma comparativa

Nas simulações até o momento executadas, ainda não foi possível de fato comprovar a eficiência ou não do método DPR e nem definir um padrão de comportamento do processo evolutivo durante as execuções, devido à pequena quantidade de testes realizados. Isto pode ser verificado nas descrições dos quatro testes realizados.

a. Teste "Um"

Este teste esta relacionado na tabela 3 como teste "07", onde foram estabelecidos os seguintes valores para os parâmetros: gerações = 5000; partidas = 100; T = 30; R = 25; P = 15; S = 10.

Os resultados obtidos para essa execução apresentaram o melhor indivíduo com um valor de fitness igual a 23.284 Km, na geração 761 e, ao final de todas as gerações, apresentou um total de 291.276 indivíduos iguais ao melhor indivíduo, ou seja, 58% da população homogênea, o que indica a perda da diversidade nessa população e a conseqüente estagnação de seu processo de evolução.

Além disso, é interessante citar que a configuração final das estratégias de comportamento entre os 100 indivíduos da última geração apresentou a estratégia *ALL*-C predominantemente, o que à princípio vai de encontro com a característica do jogo Dilema do Prisioneiro, que força naturalmente a traição mútua dos jogadores, que é o único Equilíbrio de Nash presente no jogo.

Veja na figura 7 a seguir o processo de evolução para este teste, e observe que inicialmente ocorre uma grande evolução e, posteriormente, uma evolução menor ocorre (área demarcada na figura) e depois há a estagnação do processo evolutivo da população, considerando-se o melhor indivíduo.

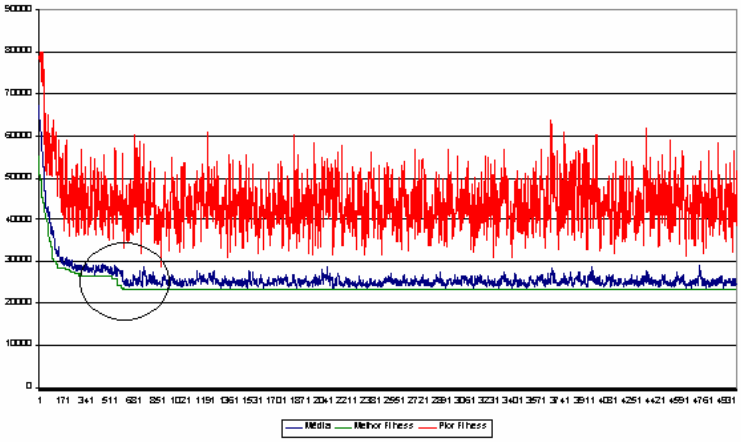


Figura 7: Melhor e pior indivíduos, média de fitness para o teste "Um"

b. Teste "Dois"

Este teste esta relacionado na tabela 3 como teste "08", onde foram estabelecidos os seguintes valores para os parâmetros: gerações = 10000; partidas = 100; T = 30; R = 25; P = 15; S = 10.

O melhor indivíduo encontrada nesta execução foi o pior entre os quatro testes realizados para o método DPR e entre todos os dez testes realizados e apresentados na tabela 3. Sendo assim, o melhor indivíduo apresentou uma fitness igual a 28.605 Km, na geração 817, e, ao final da execução, apresentou 701.684 indivíduos idênticos ao melhor indivíduo, ou seja, 70,1% da população, o que indica uma rápida perda da diversidade e convergência para um mínimo local.

A configuração final das estratégias de comportamento para a última geração desta execução apresentou todos os indivíduos portadores da estratégias ALL-D, e está de acordo com a natureza do jogo Dilema do Prisioneiro.

A seguir, na figura 8, é apresentado o gráfico de evolução para este teste. E, da mesma forma que o anterior, é interessante observar a grande evolução inicial e uma pequena evolução que ocorre na seqüência (área demarcada na figura) e, posteriormente, a estagnação completa do processo evolutivo.

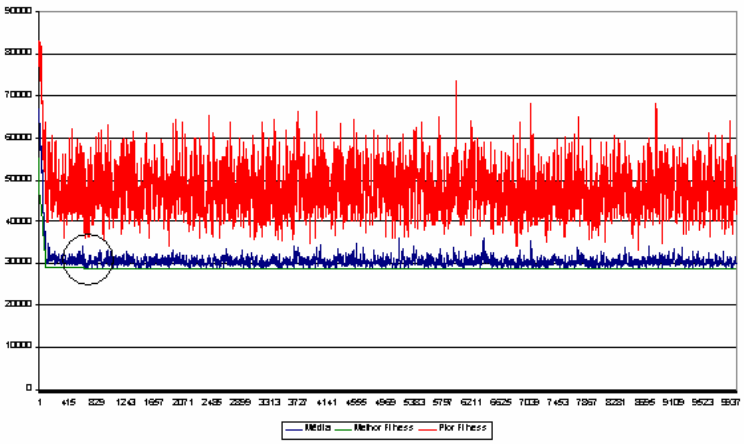


Figura 8: Melhor e pior indivíduos, média de fitness para o teste "Dois"

c. Teste "Três"

Este teste esta relacionado na tabela 3 como teste "09", onde foram estabelecidos os seguintes valores para os parâmetros: gerações = 5000; partidas = 1000; T = 30; R = 25; P = 15; S = 10.

O melhor indivíduo para este teste apresentou um valor de fitness igual a 25.433 Km, somente na geração 3.102, e apesar disso, ao final da execução, foram gerados 143.644 indivíduos idênticos, ou seja, 28,7% da população. Porém, considerando-se somente a partir do seu primeiro surgimento na população temos uma porcentagem equivalente a 75,6% do restante da população, indicando mais uma vez a estagnação do processo de evolução da população.

Já a configuração final das estratégias de comportamento apresentou-se com totalidade de indivíduos portadores da estratégia *ALL*-D, como esperado.

A seguir, na figura 9, podemos visualizar a grande evolução inicial, o que é comum, e posteriormente, nas duas áreas demarcadas, onde na primeira, ocorreu uma evolução em apenas uma única geração, com a conseqüente estagnação do processo evolutivo e, mais adiante, na geração 3.102, uma evolução mais forte ocorreu, conforme pode ser visto na segunda área demarcada na figura.

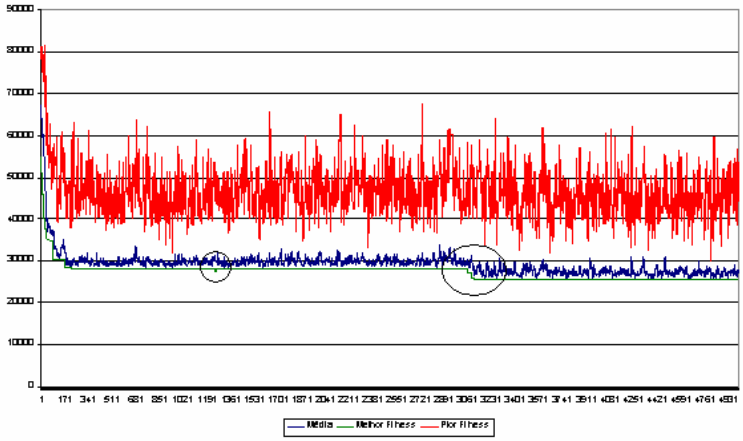


Figura 9: Melhor e pior indivíduos, média de fitness para o teste "Três"

d. Teste "Quatro"

Este teste esta relacionado na tabela 3 como teste "10", onde foram estabelecidos os seguintes valores para os parâmetros: gerações = 10000; partidas = 1000; T = 30; R = 25; P = 15; S = 10.

Neste teste foi encontrado o melhor indivíduo para os quatro testes realizados para o método DPR, com um valor de fitness igual a 22.586, na geração 474, e com 70,9% da população idêntica, ao final das dez mil gerações, indicando assim, a forte perda da diversidade ocorrida.

A configuração final das estratégias de comportamento, da última geração da população, demonstra que todos os indivíduos apresentaram-se portadores da estratégia *ALL*-D, o que já era esperado.

A seguir, na figura 10, podemos ver o único grande processo evolutivo, que ocorreu no início da execução.

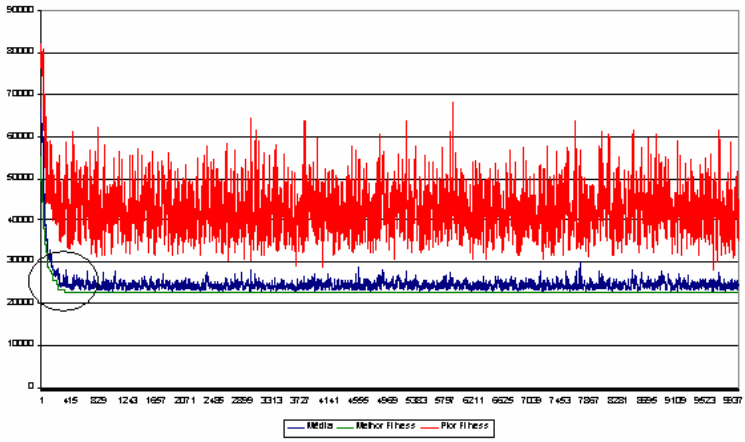


Figura 10: Melhor e pior indivíduos, média de fitness para o teste "Quatro"

7. CONSIDERAÇÕES FINAIS

Da mesma forma como apresentada em [5], o uso da Teoria dos Jogos para otimizar os Algoritmos Genéticos ainda é uma área pouco estudada e desenvolvida, e que ainda há muito a ser pesquisado, pois a combinação dos diversos métodos de seleção, com os diversos jogos, abrem um vasto campo de possibilidades, e que precisam ser exploradas. E é o que este trabalho ajuda a iniciar.

Para o Problema do Caixeiro Viajante Simétrico tido como problema teste para o método em questão, este obteve resultado razoável para os valores dos parâmetros definidos nas simulações, não podendo ainda afirmar se o método DPR contribui de forma efetiva para a solução do PVCS.

As duas contribuições apresentadas neste trabalho são em relação a: uma primeira simulação do uso do jogo Dilema do Prisioneiro no contexto dos Algoritmos Genéticos conforme os objetivos aqui propostos, permitindo aos indivíduos da população disputarem recursos disponíveis em um meio ambiente social, representado através do Dilema do Prisioneiro, e, segundo, com as execuções realizadas foi obtido um melhor resultado para o melhor indivíduo, apesar deste ter sido encontrado utilizando-se o método HDR.

Um fato importante e que deve ser evidenciado é que dentre os dez testes realizados, o décimo teste, este com o método de seleção DPR, apresentou o segundo melhor resultado, o que pode ser um indicativo para acreditarmos que a inclusão do jogo Dilema do Prisioneiro, antes da etapa de reprodução, possa de fato auxiliar na busca de melhores resultados, assim, melhorando a eficiência do Algoritmo Genético, uma vez que ajuda a diminuir o grau de aleatoriedade do processo de otimização, a partir do momento em que mais um fator é considerado para o processo de busca.

Em comparação com os valores obtidos em [12] e [4], 20.490 Km e 20.113 Km, respectivamente, para o mesmo método de seleção HDR, com as simulações realizadas para comparar com os resultados do método DPR, foi possível encontrar um valor ainda melhor, isto é, 20.093 Km e, mesmo assim, ainda não se pode afirmar que este valor represente o mínimo global para o problema exposto, ou seja, que esta é a menor distância entre as 26 capitais brasileiras que se comunicam mutuamente via rodovias.

Referências

[1] BEASLEY, D.; BULL, D. R.; MARTIN, R. R. An Overview of Genetic Algorithms: Part 1, Fundamentals. Inter-University Committee on Computing. University Computing, UK, 1993.

[2] BORGES, P. S. S. A Model Of Strategy Games Based on The Paradigm of the Iterated Prisioner's Dilemma Employng Fuzzy Sets. 1996. Tese (Doutorado em Engenharia de Produção) – Universidade Federal de Santa Catarina, Florianópolis.

[3] DARWIN, C. Origem das espécies. Tradução do original "On the origin of species: by means of natural selection, or preservation of favoured races in the struggle for life", feita por Eugênio Amado. Belo Horizonte: Villa Rica, 1994.

[4] DE BRITO, F. H. Hawk-Dove Torneio: um novo método de seleção para os algoritmos genéticos baseado na teoria dos jogos evolucionários e estratégias evolucionárias. 143f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro Universitário do Estado do Pará, Belém, 2004.

[5] DE BRITO, F. H.; TEIXEIRA, O. N.; DE OLIVEIRA, R. C. L. A Introdução da Interação Fenotípica em Algoritmos Genéticos através dos Jogos Evolucionários e da Codificação e Transmissão Genética do Comportamento, 2005. (Submetido ao VII Simpósio Brasileiro de Automação Inteligente. 2005 Latin American IEEE Robotics Symposium).

[6] DNIT. Departamento Nacional de Infra-estrutura de Transportes. Disponível em: http://www.dnit.gov.br/. Acessado em: 01. nov. 2004.

[7] EBERHART, R.; SIMPSON, P.; DOBBINS, R. Computational intelligence PC tools: an indispensable resource for the latest in fuzzy logic, neural network and evolutionary computing. American Press Inc., 1996.

[8] FIANI, R. Teoria dos Jogos: Para Cursos de Administração e Economia. 3ª Reimpressão. Rio de Janeiro: Elsevier, 2004.

[9] FOGEL, D. B. Evolutionary computation: toward a new philosophy of machine intelligence. 2.ed. IEEE Press, 2000.

[10] GOLDBERG, D. Genetic Algorithms in search, optimization and machine learning. Addison Wesley Longman, Inc., 1989.

[11] HAUPT, R. L.; HAUPT, S. E. Practical genetic algorithms. John Wiley & Sons, Inc., 2000.

[12] LEHRER, C. Operador de seleção para algoritmos genéticos baseado no jogo hawk-dove. 2000.123f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2000.

[13] LUCE, D. R.; RAIFFA, H. Games and decision: introduction and critical survey. New York: Dover, 1957.

[14] MITCHELL, M. An introduction to genetic algorithms. MIT Press, 1999.

[15] TEIXEIRA, O. N. Computação evolucionária: dos aspectos filosóficos à implementação dos algoritmos genéticos na solução do problema do caixeiro viajante simétrico. 267f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Pará, Belém, 2003.

[16] TEIXEIRA, O. N. Uma proposta de um novo algoritmo genético através do uso da teoria dos jogos. Dissertação de Mestrado (Programa de Pós-graduação em Engenharia Elétrica – Computação Aplicada) - Universidade Federal do Pará, Belém, 2005.

Representación y Clasificación de Datos Geoespaciales Usando Redes Neuronales

Marly Esther De Moya Amarís
Instituto Geográfico Agustín Codazzi, Subdirección de Geografía y Cartografía
Bogotá, Colombia
me_demoya@igac.gov.co
y
Luis Fernando Niño Vásquez, Ph.D.
Universidad Nacional de Colombia, Laboratorio de Sistemas Inteligentes,
Bogotá, Colombia
lfninov@unal.edu.co

Abstract

Approximately 80% of all the existing information in the world corresponds to geo-referenced information, this creates an increasing necessity to have tools more flexible, precise and easy to use to perform visualization, exploration and classification of great volumes of geospatial data. Additionally it is necessary to achieve shorter times to process this kind of information. In this work, different techniques to visualize and classify geo-referenced data using two types of neuronal networks: Kohonen's maps (SOM) and the Neural Gas method (NG) are compared. In visualization, SOM showed a better performance than NG, while in classification NG performed better than SOM.

Keywords: Neural Networks, Self Organizing Map (SOM), Kohonen, Neural Gas, Geographic Information System (GIS).

Resumen

Aproximadamente el 80% de toda la información existente en el mundo corresponde a información geo-referenciada. Esto crea una creciente necesidad de disponer de herramientas más flexibles, precisas y fáciles de usar para la visualización, exploración y clasificación de grades volúmenes de datos Geoespaciales. Adicionalmente es necesario lograr menores tiempos de procesamiento para este tipo de información. En este trabajo se comparan diferentes técnicas para presentar y clasificar datos geo-referenciados utilizando dos tipos de redes neuronales: mapas auto-organizativos de Kohonen (SOM) y el método gas neuronal (GN). Para los casos de visualización, SOM mostró un mejor desempeño que GN, dándose el caso contrario para los ejemplos de clasificación.

Palabras Clave: Redes Neuronales, Mapas auto-organizativos (SOM), Kohonen, Gas Neuronal, Sistemas de Información Geográfica (GIS)

1 INTRODUCCIÓN

Debido a la creciente cantidad de información georreferenciada, se tienen que almacenar, procesar y analizar volúmenes de información cada día más grandes. Con el fin de manejar esta enorme cantidad de datos complejos y multidimensionales, se ha incrementado la demanda de técnicas más sofisticadas de análisis, incluyendo clasificación y visualización de información geoespacial. Tradicionalmente, los sistemas de información geográfica (GIS) han sido ampliamente utilizados para analizar y visualizar datos geoespaciales. En los últimos años, una nueva generación de sistemas de información geográfica han surgido y han extendido sus funcionalidades, generando mapas dinámicamente, logrando grandes capacidades de análisis exploratorio visual de los datos [1], [3], [5], [9]. A pesar de este gran desarrollo, estos sistemas tienen capacidades limitadas para visualizar la interacción de los atributos en mapas con apenas unas pocas dimensiones, de aquí, que dependencias complejas multivariadas sean fácilmente pasadas por alto, teniendo como consecuencia que mucho conocimiento valioso oculto en los datos probablemente nunca sea descubierto o presente mucha dificultad de ser hallado.

En esta investigación se estudia y se demuestra el aporte que las Redes Neuronales Artificiales pueden proveer en este campo, ya que éstas presentan grandes ventajas que pueden ser aplicadas al estudio de datos geoespaciales, tales como sus capacidades de clasificación, reorganización topológica, reducción de datos y visualización. El estudio se centró específicamente en un tipo de redes neuronales conocido como mapas auto-organizativos, básicamente se trabaja con

dos clases de mapas: los mapas de Kohonen (SOM) [7] y el método gas neuronal (GN) [8]. Estas técnicas presentan grandes ventajas para analizar conjuntos de datos masivamente complejos y relacionados espacialmente. El desarrollo de ambientes de visualización basados en mapas auto-organizativos pretende descubrir estructuras y patrones en complejos conjuntos de datos espaciales y proveer reducción de datos y representaciones gráficas que puedan soportar el procesamiento, el análisis, la comprensión y la construcción de conocimiento. La capacidad de estas redes se mostró usando datos del censo de Colombia del año 1993, utilizando tanto el método SOM como el GN para su clasificación, representación y visualización, se estudió cada método en detalle y después se realizó un análisis comparativo de los resultados obtenidos con cada método. Un estudio de las mismas características se realizó para la visualización de modelos digitales del terreno.

El resto de este artículo está organizado como sigue: en la sección 2 se presenta una descripción de los mapas auto-organizativos de Kohonen y del método Gas Neuronal, para posteriormente, en la sección 3, presentar el marco experimental, describiendo los conjuntos de datos utilizados en los análisis, así como su correspondiente preprocesamiento. En la sección 4 se analizan los resultados obtenidos tanto para visualización como para clasificación utilizando ambas técnicas. En la sección 5, se analizarán algunos indicadores, a fin de comparar la eficiencia de los SOM y del método NG en la generación de vectores prototipos como representantes del conjunto completo de datos. Finalmente, en la sección 6 se resume el trabajo realizado y se presentan las conclusiones finales

2 MAPAS AUTO-ORGANIZATIVOS Y GAS NEURONAL

Los mapas auto-organizativos y el método gas neuronal corresponden a ejemplos de redes competitivas, en las cuales se define algún tipo de competición entre unidades con el fin de conseguir que una de ellas quede activada. Esto se consigue mediante aprendizaje no supervisado, presentando algún patrón de entrada y seleccionando la unidad cuyo patrón de pesos incidentes se parezca más al patrón de entrada, lo cual genera el refuerzo de dichas conexiones y de sus vecinas, y debilitando el de las demás unidades perdedoras. Al final se consigue que cada unidad responda frente a determinados patrones de entrada, y las neuronas vecinas se activarán de manera que los pesos aferentes de esa unidad converjan en el centro del grupo de patrones con características similares. Para este caso de estudio, se analizarán estos dos tipos de redes neuronales competitivas: los mapas auto-organizativos de Kohonen y el método Gas Neuronal.

2.1 Mapas auto-organizativos de Kohonen

El aprendizaje del modelo de Kohonen es no supervisado de tipo competitivo. Las neuronas de la capa de salida compiten por activarse y sólo una de ellas permanece activa ante una determinada información de entrada a la red. Los pesos de las conexiones se ajustan en función de la neurona de haya resultado vencedora.

El algoritmo de aprendizaje utilizado para establecer los valores de los pesos de las conexiones entre las N neuronas de entrada y las M de salida es el siguiente:

1. Inicializar los pesos (w_{ij}) con valores aleatorios pequeños y fijar la zona inicial de vecindad entre las neuronas de salida.
2. Presentar una entrada en forma de vector $E_k = (e_1^{(k)},...., e_N^{(k)})$, donde los componentes $e_i^{(k)}$ serán números reales.
3. Determinar la neurona vencedora de la capa de salida, la cual será la que tenga el valor más parecido al patrón de entrada E_k. Para ello, se calculan las distancias o diferencias entre ambos vectores, considerando una por una todas las neuronas de salida:

$$d_j = \sum_{i=1}^{8} (e_i^{(k)} - w_{ij})^2 \quad 1<=j<=M \text{ , donde} \tag{1}$$

$e_i^{(k)}$: Componente i-ésimo del vector k-ésimo de entrada.

w_{ij}: Peso de la conexión entre la neurona i de la capa de entrada y la neurona j de la capa de salida.

4. Una vez localizada la neurona vencedora (j^*), se actualizan los pesos de las conexiones entre las neuronas de entrada y dicha neurona, así como los de las conexiones entre las de entrada y las neuronas vecinas de la vencedora.

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t)\left[e_i^{(k)} - w_{j^*i}(t)\right] \text{ , donde} \tag{2}$$

$\alpha(t)$: Parámetro de ganancia o coeficiente de aprendizaje, con un valor entre 0 y 1, el cual decrece con cada iteración.

5. El proceso se debe repetir, volviendo a presentar todo el juego de patrones de aprendizaje.

El mapa construido (SOM) es una representación plana de los vectores prototipo, imaginados como puntos localizados en el espacio de datos. La eficiencia de esta representación es medida por dos índices:

1. *Error de Cuantización*. Este error corresponde al promedio de la distancia Euclidiana de los vectores de datos a sus representantes más cercanos.
2. *Error Topológico.* Este error indica cual es la fracción de vecinos en el mapa, los cuales no tienen regiones de Voronoi en el espacio de datos.

2.2 El método gas neuronal

El método GN, propuesto por Martinetz [2], se orienta hacia la cuantización del espacio. Este método es más general comparado con los mapas de Kohonen. La mayor diferencia entre los métodos SOM y GN, es que en el modelo GN la colección de neuronas no está conectada por una red: cada neurona puede moverse libremente a través del espacio de datos. Las coordenadas de las neuronas son llamadas tradicionalmente "pesos". Si se considera una colección de *m* neuronas, cada neurona puede ser imaginada como un punto en el espacio de datos. Las coordenadas de la $i^{ésima}$ neurona será denotada como $w_i = (w_{i1}, ...,w_{ip})$. Al inicio de los cálculos, la colección de neuronas es distribuida aleatoriamente sobre el espacio de datos. En las siguientes iteraciones, las neuronas cambian su posición y se adaptan ellas mismas a la nube de datos. El proceso de adaptación es llamado *aprendizaje* o *entrenamiento*. El entrenamiento es realizado en ciclos llamados épocas. Durante cada época *n* vectores de datos (elegidos en orden aleatorio del espacio de entrada) son presentados secuencialmente al conjunto de neuronas. Asumiendo *h* épocas, el máximo número de iteraciones de ajuste permitidas es K_{max}= n * h. En cada iteración *k* (*k*=0,1,.... K_{max}), se presenta un vector de datos aleatorio **x** al conjunto de neuronas. Para cada vector de datos **x** presentado en la k-ésima iteración se encuentra la neurona más cercana (cercana en el sentido Euclidiano). Esta neurona es llamada ganadora y obtiene el índice *w* El vector de pesos de la neurona ganadora satisface la siguiente relación:

$$d(x, w_w) = \min\ d(x, w_i) \quad 1<=i<=m \tag{3}$$

En el paso siguiente se establece el vecindario de la neurona ganadora. La magnitud (diámetro) del vecindario decrece exponencialmente con (*k*), el número actual de la presentación. Para cada *k* todas las neuronas pertenecientes al vecindario de la neurona ganadora cambian su posición para ubicarse más cerca del vector **x** actualmente expuesto. El cambio es descrito por la fórmula:

$$w_i = w_i + \alpha(k)G(i,k,x,w,\lambda)(x - w_i) \tag{4}$$

donde la función G describe el vecindario de la neurona ganadora, es decir, la vecindad del vector $\mathbf{w}_w$:

$$G(i,k,x,w,\lambda) = \exp\left\{-\frac{d^2(x,w_i)}{2\lambda^2(k)}\right\} \tag{5}$$

y el índice *i* toma valores sobre todas las neuronas pertenecientes al vecindario establecido para la ganadora *w*.

Analizando la fórmula (4) puede mirarse que los cambios de posición de la *i*-ésima neurona depende de dos factores: α, y el valor de la función G, el cual depende del parámetro λ. Ambos coeficientes α y el parámetro λ depende de *k*, el número de la actual iteración, teniendo el siguiente significado:

α(*k*) : Define el coeficiente de aprendizaje, el cual determina qué tan grande puede ser el cambio de posición. El coeficiente α usualmente decrece con el número de iteraciones: con un valor de inicio α_0, decreciendo gradualmente a un valor final α_{min} alcanzado al final de todas las iteraciones. El valor de decrecimiento de α(*k*) es descrito por la función:

$$\alpha(k) = \alpha_0\left(\frac{\alpha_{min}}{\alpha_0}\right)^{k/k_{max}} \tag{6}$$

λ(*k*): Define el diámetro del área de vecindad en la *k*-ésima presentación. Normalmente esta decrece con *k*, el número actual de presentación: para un valor inicial λ_0, decreciendo gradualmente hasta un valor final λ_{min} al final de las presentaciones para $k = k_{max}$:

$$\lambda(k) = \lambda_0\left(\frac{\lambda_{min}}{\lambda_0}\right)^{k/k_{max}} \tag{7}$$

Puede decirse que la fórmula anterior expresa la adaptación de la neurona ganadora y sus vecinas en la dirección del vector de datos **x** presentado.

3 MARCO EXPERIMENTAL

El objetivo general del estudio experimental es realizar un análisis comparativo entre los mapas auto-organizativos de Kohonen (SOM) y el método Gas Neuronal (GN), aplicados a la clasificación y visualización de datos geoespaciales, a fin de evidenciar cual presenta un mejor desempeño en estas tareas. El estudio está dividido en dos tipos de análisis:

- *Clasificación de datos geoespaciales y su correspondiente visualización y representación.* El objetivo de este análisis es realizar un estudio comparativo entre los mapas auto-organizativos de Kohonen y el método gas neuronal, aplicados a la clasificación y representación de la información socioeconómica y demográfica de la población de Colombia, georeferenciada a nivel municipal, correspondiente al censo de 1993. La fuente de datos es el Departamento Nacional de estadística – DANE [2]. Esta muestra incluye los siguientes factores: actividad económica, población por sexo y edad, nivel de educación, ocupación, condición de tenencia de la vivienda, tipo de vivienda, material de las paredes, material del piso, servicios públicos, servicio sanitario, en que lugar cocinan, con qué cocinan. Conformando un total de 118 variables por cada uno de los 1029 municipios.

- *Visualización de datos geoespaciales.* El objetivo es realizar un estudio comparativo entre los mapas auto-organizativos de Kohonen y el método gas neuronal, aplicados a la visualización tridimensional del terreno (generación de modelos digitales del terreno). Para este análisis se seleccionó una muestra de datos tridimensionales de una zona montañosa de Colombia (65535 coordenadas *x, y, z*), véase figura 1. Esta zona corresponde a un área de 38.756 Km^2, con coordenadas: esquina superior derecha: 834.969 mE, 1319476 mN, esquina inferior izquierda: 1'029.969 mE, 1'120.726 mN. La información fue adquirida en el Instituto Geográfico Agustín Codazzi de Colombia. El objetivo es analizar la capacidad de cada uno de los métodos, mapas auto-organizativos de Kohonen y el método gas neuronal, para modelar el terreno, es decir, para presentar visualmente la forma del relieve.

Figura 1: Zona de trabajo

Se comparará la eficiencia de cada uno de los dos métodos (SOM y GN), tanto para visualización como para clasificación de datos geoespaciales, utilizando el software SOMTOOLBOX para Matlab 5.3.

3.1 Preprocesamiento de datos

El primer paso es construir los conjuntos de datos y dejarlos en un formato apropiado para su procesamiento. Para facilitar el análisis de las visualizaciones generadas por los mapas auto-organizativos, se dividió el conjunto de datos por departamentos, por tanto se generó un conjunto de datos por cada uno de ellos, con todas las variables generadas en el censo de 1993 para personas, hogares y viviendas (118 variables en total), agregando las correspondientes coordenadas *x* y *y* a cada municipio. Los valores fueron traducidos a porcentajes, ya sea de viviendas, hogares o personas de acuerdo al caso, a excepción de la población total, población de mujeres y población de hombres.

Se dispone de la información de un total de 1029 municipios, en la figura 2 se puede apreciar la distribución geográfica de los mismos.

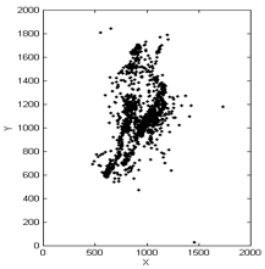


Figura 2. Distribución de municipios

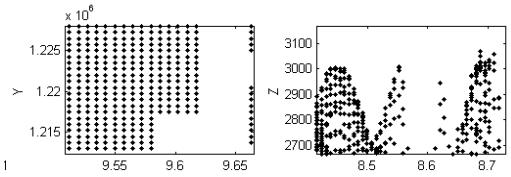


Figura 3. Acercamiento de la malla de puntos

Para el caso del modelo digital del terreno se generaron 16 archivos, 14 con 4096 puntos y 2 con 4095 puntos, con coordenadas *x, y* y *z*, por tanto, en total se dispone de una malla de 65535 puntos. En la figura 3 se muestra un acercamiento al conjunto de puntos.

Una vez los conjuntos de datos han sido generados y almacenados en su correspondientes formato, se procede a la normalización. Debido a que los algoritmos SOM y GN están basados en las distancias existentes entre los datos, la escala de las variables es muy importante en la determinación de los vectores prototipo. Si el rango de una variable es mucho más grande que el de las otras, probablemente, esta variable dominará la organización de los resultados completamente. Por esta razón, los componentes de los datos deben ser normalizados. El proceso de normalización llevado a cabo es por rango, es decir, se toma el valor máximo y mínimo de cada conjunto de datos y se realiza una transformación lineal la cual escala los valores entre [0, 1].

4 RESULTADOS EXPERIMENTALES

A continuación se describen los resultados obtenidos con los experimentos. Para analizar el comportamiento del GN y del SOM se aplicaron diferentes parámetros sobre cada conjunto de datos, a fin de evaluar el comportamiento y determinar los valores óptimos de desempeño.

4.1 Clasificación, representación y visualización de los datos del censo

Para realizar el entrenamiento de las redes se seleccionó una muestra de 10 departamentos, de un total de 30, tal como se ilustra en la tabla 4.1. En esta muestra se incluyen departamentos representativos de cada una de las regiones geográficas de Colombia.

REGION	DEPARTAMENTO	CARACTERÍSTICA
NORTE	ATLÁNTICO	Puerto aéreo, marítimo y fluvial
	BOLIVAR	Turismo
	ANTIOQUIA	Mayor número de municipios, comercio, agricultura, industria
PACIFICO	CHOCO	Pobreza
	VALLE	Puerto marítimo, agricultura, industria, comercio
CENTRO	BOYACA	Agricultura, artesanías
	CUNDINAMARCA	Capital de Colombia, agricultura
ZONA CAFETERA	CALDAS	Agricultura, comercio
LLANOS ORIENTALES	META	Agricultura
AMAZONIA	CAQUETA	Zona selvática de Colombia

Tabla 1: Departamentos seleccionados

Cada departamento cuenta con información de las 116 variables mencionadas en la sección 3 y con sus respectivas coordenadas planas *x* y *y,* las cuales permiten ubicar geográficamente cada municipio. Debido a que se dispone de un total de 116 variables por cada uno de los diez departamentos, a fin de facilitar el proceso de análisis, se seleccionó un conjunto de 53 de éstas, con el objeto de realizar un estudio del nivel de socioeconómico de la población. Este análisis pretende descubrir las interrelaciones existentes entre las diferentes variables, tales como su nivel de educación, la actividad económica, la ocupación, la condición de tenencia de vivienda, el material del piso, material de las paredes y los servicios públicos.

Se realizó el entrenamiento de una red neuronal por cada uno de los diez departamentos seleccionados, por los métodos de Kohonen y de gas neuronal. La red recibe como entrada las coordenadas geográficas de cada uno de los municipios que conforman el respectivo departamento y el conjunto de las 53 variables.

4.1.1 Desempeño con SOM

El proceso de entrenamiento de los SOM para los diez departamentos escogidos generó los resultados relacionados en la tabla 2. Aquí puede observarse, el tamaño del SOM seleccionado para cada departamento tiene un número de neuronas mayor que el número de municipios (ejemplos de entrenamiento), por esta razón los errores topográficos y de cuantización son bastante pequeños. Esto puede generar sobreentrenamiento en la red, pero debido a que el objetivo del análisis es utilizar las ventajas del despliegue gráfico de los mapas auto-organizativos de Kohonen, una red grande brinda muchas ventajas para el agrupamiento (clusters) y el análisis visual.

Departamento	No. de Munic.	SOM		GN	
		No. de Neuronas	Error de Cuantización	No. de Neuronas	Error de Cuantización
Antioquia	124	1944	0.0260	1240	0.0071
Atlántico	23	384	0.0020	230	4.8348e-004
Bolívar	32	448	0.0104	320	0.0011
Boyacá	123	2016	0.0152	1230	0.0105
Caldas	25	400	0.0026	250	6.8099e-004
Caquetá	15	180	0.0082	150	1.3502e-004
Chocó	19	216	0.0196	190	3.0326e-004
Cundinamarca	115	1980	0.0115	1150	0.0074
Meta	29	448	0.0066	290	0.0011
Valle	42	560	0.0272	420	0.0019

Tabla 2: Cuadro comparativo entrenamientos SOM y GN – Departamentos

Para un análisis más detallado, se seleccionó el departamento de Caldas, por tener buenas características de agrupamiento, facilitando esto la explicación de los mecanismos de análisis que brindan los SOM. El departamento de Caldas cuenta con 25 municipios distribuidos geográficamente tal y como se muestra en la figura 4, se seleccionó un tamaño para la red neuronal de 20 x 20 neuronas, generando un mapa de 400 neuronas (figura 5).

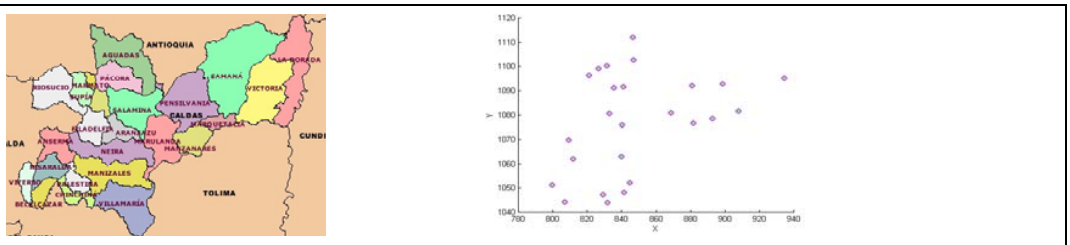


Figura 4: Ubicación geográfica de los municipios del Departamento de Caldas.

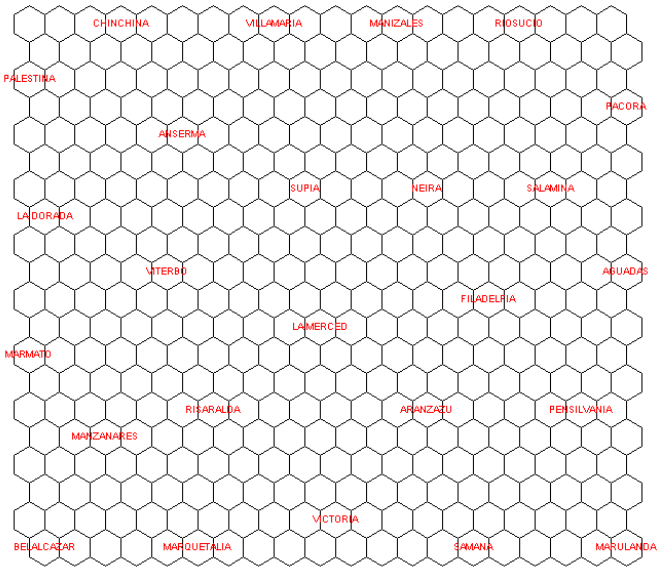


Figura 5: SOM generado para el departamento de Caldas

En la figura 5, se pueden apreciar cada uno de los municipios y demás vectores prototipo creados (neuronas o unidades de mapa), en el espacio de proyección del SOM que corresponde a la malla de la red neuronal. Sin embargo, los datos de entrada y los vectores formados también pueden ser visualizados en el espacio geográfico correspondiente (ver figura 6). Las cruces negras corresponden a cada uno de los vectores prototipo generados durante el entrenamiento y cada uno de los puntos en otros tonos corresponden a los datos de entrenamiento de la red. Esta red generó un error de cuantización de 0.0026 y un error topográfico de 0.

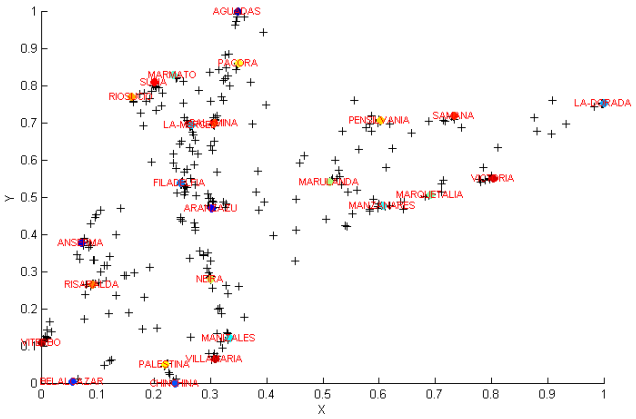


Figura 6: Datos de entrada y vectores prototipo en el espacio geográfico de Caldas

Los SOM ofrecen grandes ventajas de visualización como son las matrices de distancias y las componentes planas. Cada componente plana muestra los valores de una variable en cada unidad de mapa usando la misma codificación de color descrita para la matriz de distancia. Esto da la posibilidad de examinar visualmente cada celda (correspondiente a cada unidad de mapa). En la figura 7 se pueden apreciar las componentes planas y la matriz de distancias de las variables correspondiente al material en que están construidas las casas de Caldas.

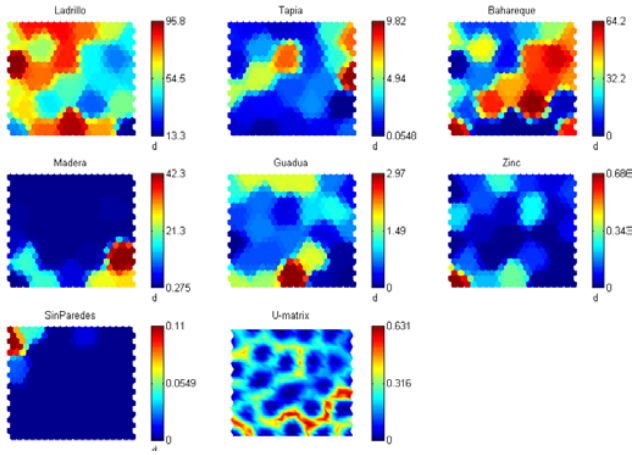


Figura 7: Material de las casas del departamento de Caldas

Aquí puede apreciarse que en los municipios de Manizales, Chinchiná, Villamaría, Supia, Viterbo, La Dorada, Marquetalia, Victoria y Samaná, se visualiza un gran porcentaje de casas construidas en ladrillo, con porcentajes que oscilan alrededor del 90%, aunque un alto porcentaje de población aún tiene casas construidas con bahareque, tal como es el caso de los municipios de Belalcazar, Marulanda, Risaralda, Aranzú, La Merced, Filadelfia, Aguadas, Neira, Salamina y Pácora, en los cuales las casas de bahareque oscilan entre el 50 y el 64%. Si se forma una matriz de distancia (U-matrix en Figura 7) con estas variables, se visualizan básicamente dos grupos, el formado por las casas de ladrillo y el formado por las casas de bahareque y un pequeño tercer grupo formado por las casas de madera (42.3%) del municipio de Pensilvania.

Una pregunta interesante es en qué parte de un SOM se encuentra localizado un dato de entrada específico, y cuanta precisión tiene esa localización. La forma más sencilla de resolver esta pregunta es encontrando la unidad del mapa que más concuerde con el vector del dato que se está buscando (BMU). Como ejemplo se van a localizar en el SOM las BMU de dos municipios, Manizales y Belalcazar (indicadas por etiquetas en el mapa), luego se van a calcular los errores relativos de cuantización, el error de cuantización entre estos ejemplos y todas las unidades del mapa. Esto puede visualizarse en la figura 9.

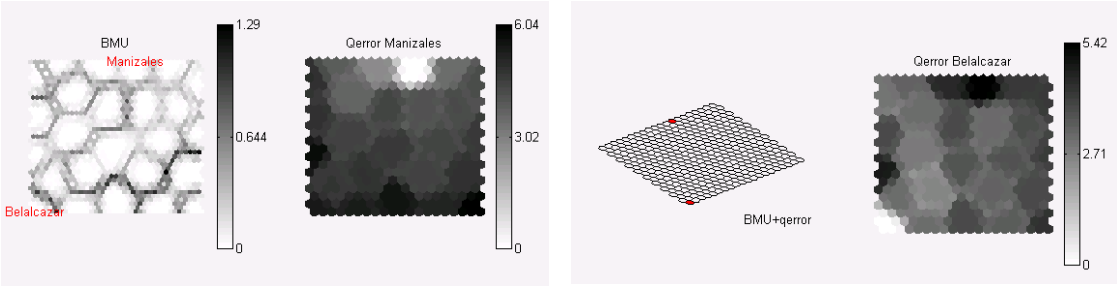


Figura 9: Localización de datos en el SOM y su precisión

El error de cuantización calculado para el municipio de Manizales, es decir la distancia hasta su BMU es de 0.0019, a su segunda BMU es de 0.007 y a su peor unidad WMU es de 6.0446. Los mismos errores se calcularon para el municipio de Belalcazar, obteniéndose 0.0011, 0.0077 y 5.4205 respectivamente.

4.1.2 Desempeño con GN

Los parámetros para el entrenamiento fueron inicializados con los valores: *alpha=0.5* y *lambda* =n/2. Los valores de $\alpha(k)$ y G(.) de la ecuación (1) declinan en iteraciones sucesivas de acuerdo a las fórmulas (3), (2) y (4). Los valores α_{min} y λ_{min} de las ecuaciones (3) y (4) toman sus valores por defecto: α_{min}=0.005 y λ_{min}=0.01.

El entrenamiento se realizó sobre los mismos diez departamentos trabajados con SOM. Los datos de entrenamiento y los errores generados se encuentran relacionados en la tabla 2. El número de neuronas seleccionadas para cada departamento es el número de ejemplos de entrenamiento (número de municipios) multiplicado por un factor de diez. El número de épocas de entrenamiento es de 250. Estos valores fueron escogidos debido a que ofrecen un menor error de cuantización. Debido a que el método Gas Neuronal no trabaja comportamiento topológico, solo se maneja el error de cuantización.

Para un estudio más detallado, al igual que en el caso de SOM, se seleccionó el departamento de Caldas, a fin de poder comparar los dos métodos para el mismo conjunto de datos. La red generada para el método GN consta de 250 neuronas, a diferencia del SOM, esta red no está conformada por filas y columnas, ni tiene una topología específica, simplemente es una red constituida por 250 neuronas que pueden viajar libremente por el espacio de datos. Cuando la red es entrenada, se generan una serie de vectores prototipo, los cuales son representantes de los datos reales, es decir, los datos de entrenamiento (los 25 municipios). En la figura 10 se pueden apreciar cada uno de los municipios y demás vectores prototipo creados (neuronas). Esta visualización está referenciada en el espacio libre de los datos, es decir en el espacio geográfico, esto debido a que el método Gas Neuronal trabaja sobre el espacio de los datos de entrenamiento.

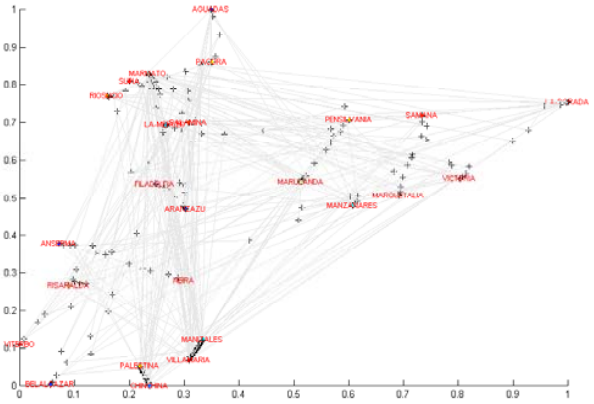


Figura 10: Datos y vectores prototipo en el espacio geográfico de Caldas

Se obtuvo un error de cuantización de 6.8099e-004. En la figura 10 puede observarse que los vectores prototipo tienden a agruparse en los sectores donde están ubicados los datos de entrenamiento y las áreas donde no existen municipios no se crearon neuronas, o se crearon muy pocas.

En un método como el GN que no tiene herramientas de visualización gráfica, las proyecciones constituyen una herramienta de gran utilidad para los procesos de análisis. A fin de facilitar el análisis visual se generó la proyección PCA para el conjunto de vectores prototipo creados durante el proceso de entrenamiento, en la figura 11 puede apreciarse la distribución de los vectores y las etiquetas indicando el nombre del municipio. Al igual que en el estudio realizado con SOM se tomaron específicamente las variables correspondientes al material de la casas. Se pueden distinguir claramente dos grupos, el primero abarcando los municipios de Pensilvania, Samaná, Manzanares, La Dorada, Marquetalia, Manizales, Chinchiná, Villamaría y un poco más alejado el municipio de Victoria. El segundo grupo puede apreciarse en la parte izquierda conformado por los municipios de Marulanda, Aguadas, Pácora, Filadelfia, Risaralda, Supia, Marmato, Viterbo, Riosucio, La Merced, Salamina, Anserma, Aranzazu, Palestina, Neira y en la parte inferior el municipio de Belalcazar. Estos dos grupos corresponden básicamente el grupo de municipios donde predomina el ladrillo y al grupo donde predomina el bahareque respectivamente, llegando de esta forma a la misma conclusión con los análisis realizados con SOM.

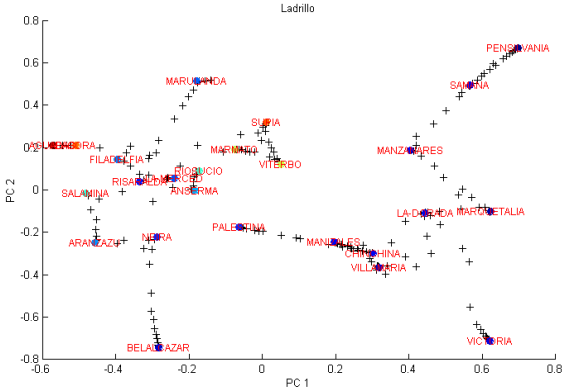


Figura 11: Proyección PCA GN – Material paredes – Caldas

4.2 VISUALIZACIÓN DE MODELOS DIGITALES DEL TERRENO

El objetivo del estudio de visualización de modelos digitales del terreno, es comparar el desempeño que ofrecen los mapas auto-organizativos de Kohonen y el método gas neuronal para representar eficientemente la forma del terreno. Como ya se explicó en la sección de descripción de los datos, se dispone de 65534 puntos, los cuales contienen coordenadas *x, y* y *z*. La red neuronal se alimentará con estos tres datos y su función será reorganizarse topológicamente y generar un modelo digital del terreno correspondiente a los puntos que ha recibo como entrenamiento. A fin de facilitar el análisis y el comportamiento de la red, se dividió el conjunto de datos en 16 regiones, 14 de ellas con 4096 puntos y 2 con 4095. Debido a que el terreno es bastante montañoso, esta división permite un análisis más detallado de cada una de las regiones.

4.2.1 Desempeño con SOM

Para el estudio del modelamiento del terreno por medio de SOM, se trabajó con 16 regiones, generando los errores de cuantización y topológicos detallados en la tabla 3. El número de puntos de cada región es de 4096 con excepción de las regiones 1-4 y 2-4 que tienen 4095 puntos. Para realizar un estudio en detalle, se seleccionó una de las 16 regiones (1-4), la cual es representativa, presentando variaciones en el detalle del relieve y diferentes alturas. En la figura 12 se muestran los vectores prototipo del área seleccionada (malla) vs. los datos reales (puntos).

Región	SOM		GN
	No. de Neuronas	Error de Cuantización	Error de Cuantización
1-1	1312	0.0230	0.0148
1-2	1302	0.0220	0.0145
1-3	1305	0.0195	0.0135
1-4	1312	0.0240	0.0147
2-1	1312	0.0190	0.0134
2-2	1290	0.0176	0.0129
2-3	1287	0.0201	0.0137

2-4	1302	0.0201	0.0135
3-1	1288	0.0163	0.0124
3-2	1290	0.0164	0.0121
3-3	1290	0.0155	0.0118
3-4	1312	0.0150	0.0109
4-1	1290	0.0211	0.0141
4-2	1312	0.0159	0.0121
4-3	1302	0.0153	0.0115
4-4	1032	0.0147	0.0114

Tabla 3: Cuadro comparativo entrenamientos SOM y GN - Modelamiento de terreno

Esta región tiene alturas entre los 185 y los 2600 metros, coordenadas mínimas 835.720 mE y 1.269.977 mN y coordenadas máximas 881.470 mE y 1319477 mN, conformando una región de 45.750 metros por 49.500 metros, es decir 2.264.625.000 m^2, alrededor de 2.265 Km^2. Los datos de entrada corresponden a 4095 puntos equidistantes entre sí, con una separación entre ellos de 750 metros tanto en la coordenada X como en la coordenada Y.

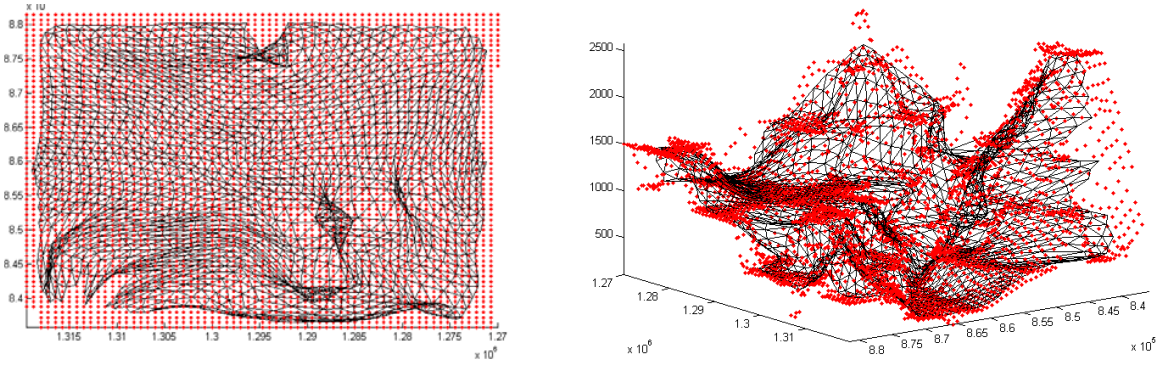


Figura 13: Vectores prototipo Vs. datos – Modelamiento de terreno con SOM

El entrenamiento del SOM permite realizar una gran reducción de la cantidad de datos a ser procesados, de los 4095 datos de entrada, se redujo a 1312 (número de neuronas), es decir, aproximadamente el 68%, esta reducción de datos generó un error de cuantización de 0.0240 y un error topográfico de 0.0256, es decir, se tiene un modelo digital del terreno con un error aproximado de 2.5%. En la figura 13 puede apreciarse la diferencia entre los datos de entrada (puntos) y los vectores prototipos creados (malla).

Como puede observarse, la malla de puntos se ajusta a los datos de entrada, aunque queda un pequeño perímetro de puntos en la parte externa de la región sin representación de vectores prototipo, a fin de obtener un mayor cubrimiento puede elegirse un mapa con mayor número de neuronas. Seleccionando un SOM de 56x44 (2464) neuronas se obtiene un error topográfico de 0.019 y un error de cuantización de 0.027. Se redujo el error de cuantización del espacio al 1.9% y se obtuvo un SOM con mayor cobertura. Comparando la tabla de errores y los SOM generados, podemos verificar que entre más homogéneo es el relieve, el entrenamiento del mapa genera errores de cuantización y topológicos más pequeños, por tanto en zonas con mayor variación de alturas y diversos tipos de relieve es aconsejable utilizar mapas con mayor número de neuronas a fin de obtener mayor precisión a la forma real del terreno y mayor cobertura al espacio de datos.

Puede concluirse que los SOM se convierten en una herramienta efectiva para generar modelos digitales del terreno, la estructura de malla propia de los SOM permite una visualización adecuada de la superficie del terreno, adicionalmente brinda la gran ventaja de reducir la cantidad de datos, modelos de relieve complicado como el usado en el análisis, permitió la reducción de datos en un 68%, con un error del 2.5% y usando más neuronas, se logró una reducción de datos en un 50% con un error del 1.9%, aumentando el tamaño del mapa se puede reducir aún más el porcentaje de error. En modelos más sencillos se puede disminuir aún más la cantidad de datos sin perder el grado de precisión, lo cual permite disminuir los tiempos de procesamiento y el espacio de almacenamiento.

4.2.2 Desempeño con GN

Para el análisis del desempeño del método GN, se trabajó con las mismas 16 regiones entrenadas con SOM, esto a fin de tener parámetros de comparación entre los dos métodos. El número de neuronas escogido para cada región (1300) es en

promedio el mismo utilizado en el entrenamiento con SOM, esto con el objetivo de analizar el comportamiento del GN con la misma cantidad de neuronas. Los errores de cuantización generados se detallan en la tabla 3.

Para el análisis detallado se seleccionó la misma región seleccionada para el estudio de SOM (1-4), esto a fin de tener parámetros de comparación entre el desempeño presentado por los dos métodos. En la figura 14 se muestran los vectores prototipo generados durante el entrenamiento del GN para el área escogida (puntos oscuros). El entrenamiento de esta región con el método GN, permitió la reducción de los datos de entrada en un 69% generando un error de cuantización del 1.4%. Para una reducción de datos tan alto, este porcentaje de error es bastante bajo. Si se desea disminuir el error obtenido pueden agregarse más neuronas a la red. Analizando la representación del relieve obtenida en la figura 14, puede observase que la adaptación al terreno del GN es muy precisa, esto debido a que este método permite que las neuronas viajen libremente por el espacio geográfico.

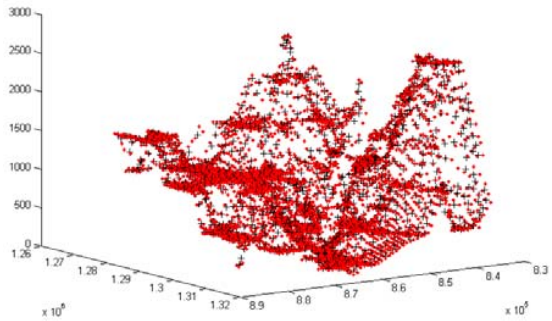


Figura 14: Vectores prototipo para modelamiento de terreno con GN

5 SOM VS. GN

Observando las tablas 2 y 3 puede observarse que el método GN con menor número de neuronas genera errores de cuantización mucho más bajos, esto es debido a que el método GN permite que sus neuronas viajen libremente por el espacio de los datos, esto facilita que las neuronas se distribuyan más eficientemente y con mayor precisión. Los SOM, por el contrario, tiene sus neuronas encasilladas en una malla y las neuronas, a pesar de que pueden moverse por el espacio de datos, tienen que respetar la malla a la que pertenecen, quitándoles esto libertad de movimiento. La gran ventaja de la malla, es que le permite a los SOM tener preservación topológica, los SOM permiten la visualización de cada una de sus neuronas y sus neuronas vecinas y respetan esta topografía, por ello los SOM manejan también el error topográfico. El método GN aunque maneja el concepto de vecindad, solo permite el manejo del error de cuantización.

De igual manera, se pueden ver las grandes ventajas que ofrece la malla de los SOM, permite diagramar las matrices de distancias y las componentes planas y facilitan la visualización de grupos y las correlaciones que existen entre las diferentes componentes planas. Los SOM ofrecen ventajas de visualización que el método GN no permite. Sin embargo, tanto los SOM como el método GN pueden combinarse con otros métodos de proyección a fin de contar con más herramientas visuales para corroborar hipótesis.

6 CONCLUSIONES

Esta investigación presentó las características de los mapas autoo-organizativos de Kohonen (SOM) y del método gas neuronal (GN) para la representación, clasificación y el análisis de datos geoespaciales. El objetivo principal fue explorar las herramientas que ofrecen estos métodos para soportar la extracción de información en grandes conjuntos de datos georreferenciados y la construcción de conocimiento a través de representaciones visuales. Se realizaron experimentos que mostraron el potencial de este tipo de redes neuronales. Del entrenamiento generado para visualización y clasificación de los datos del censo puede observarse, que el método GN con menor número de neuronas genera errores de cuantización mucho más bajos, esto es debido a que el método GN permite que sus neuronas viajen libremente por el espacio de los datos, esto facilita que las neuronas se distribuyan más eficientemente y con mayor precisión. Los SOM, por el contrario, tiene sus neuronas encasilladas en una grilla y las neuronas, a pesar de que pueden moverse por el espacio de datos, tienen que respetar la grilla a la que pertenecen, quitándoles esto libertad de movimiento. Sin embargo, la grilla ofrece grandes ventajas, permite diagramar las matrices de distancias y las

componentes planas y facilita la visualización de clusters y las correlaciones que existen entre las diferentes componentes planas.

Este trabajo mostró que las redes neuronales constituyen una herramienta de gran ayuda para el análisis, representación y visualización de datos geoespaciales, por tanto un trabajo futuro importante sería desarrollar estas aplicaciones para que puedan ser utilizadas desde los software de Sistemas de Información Geográfica existentes y así ayudar a los procesos de análisis y de visualización tradicionales.

Existen diversas aplicaciones en las cuales las redes neuronales y más específicamente los mapas auto-organizativos pueden ser de gran utilidad. Una aplicación interesante es agregar la variable tiempo a los análisis de datos georreferenciados, es decir, realizar análisis espacio-temporales de datos geoespaciales. Los mapas auto-organizativos ofrecen la posibilidad de representar los cambios ocurridos a través de tiempo mediante el trazo de trayectorias. Esto representaría una valiosa herramienta tanto para el estudio de datos geoestadísticos (como en el caso de los datos del censo), como para el estudio de modelos digitales del terreno, ya que se podrían analizar los cambios geográficos sufridos en una determinada zona a través del tiempo. De igual forma, el estudio también puede ser ampliado a información tipo *raster*, tales como imágenes de satélites y fotografías aéreas, para hacer tareas como clasificación, compresión de imágenes, procesos geoestadísticos, entre otros. Otro trabajo futuro importante sería explorar otros métodos de redes neuronales como soporte a diversas aplicaciones SIG.

Referencias

[1] Andrienko, G., Andrienko, N.. "Interactive Maps for Visual Data Exploration", *International Journal of Geographical Information Science* 13(5), 355-374, 1999

[2] Departamento Nacional de Estadística, DANE. Http://www.dane.gov.co/

[3] Dykes, J., "Exploring spatial data exploration with dynamic graphics", *Computers and Geosciences*, 23, 345-370, 1997

[4] Gahegan, M., On the application of inductive machine learning tools to geographical analysis., Geographical Analysis, Vol. 32, No 2, 113-119, 2000

[5] Gitis V., Dovgyallo A., Osher B., Gergely T., "GeoNet: an information technology for WWW on-line intelligent Geodata analysis", *Abstracts of 4th EC-GIS Workshop*, Hungary, 1998

[6] Heinke D., Hamker F.H., Comparing neural network benchmarks on growing neural gas, growing cell structure, and fuzzy ARTMAP. *IEEE Trans. on Neural Network*, pp. 1279–1291, 1998

[7] Kohonen T., Self-Organizing Maps. Springer Series in Information Sciences, 30, Berlin 1995.

[8] Martinetz M., Berkovich S., Schulten K.: 'Neural-gas' network for vector quantization and its application to time series prediction. IEEE Trans. Neural Networks, V. 4, 558–569, 1993

[9] Openshaw, S., Turton, I., Macgill, J. and Davy, J., "Putting the Geographical Analysis Machine on the Internet", in Gittings, B. (ed.) *Innovations in GIS 6,* Taylor and Francis, London, 1999

[10] Ultsch, A., Self-organizing Neural Networks for Visualization and Classification, in O.Optiz, B. Lausen and R. Klar, (Eds). Information and Classification, Berlín: Springer-Verlag, 307-313, 1999

[11] Vesanto J., Himberg J., Alhoniemi E., Parhankangas J., *SOM Toolbox for Matlab 5*. Som Toolbox team, Helsinki University of Technology, Finland, Libella Oy, Espoo, 1–54, 2000

Strength By Objective: Una nueva estrategia de asignación de fitness para Algoritmos Evolutivos Multi-Objetivos.

Guillermo T García
Universidad Católica Andrés Bello,
Depto. de Informática
Caracas, Venezuela, 1020
ggarciao@gmail.com

Gonzalo Ron
Universidad Católica Andrés Bello,
Depto. de Informática
Caracas, Venezuela, 1020
gonzalo_ron@hotmail.com

Wilmer Pereira
Universidad Católica Andrés Bello,
Depto. de Informática
Caracas, Venezuela, 1020
wpereira@ucab.edu.ve

Abstract

Nowadays, a lot of effort has been invested in determining heuristic techniques for finding better solutions to multiobjective optimizations. Among the variety of proposals, the evolutionary algorithm SPEA2 can be named for its proved superiority over other techniques in a wide range of experiments. One of its milestones lies in the *Fitness* assignment strategy, called *Strength*, which classifies the population using a niche mechanism, based in the concept of *Pareto Front*. Although this strategy is considered one of the best because of the obtained results, it presents a weakness, normally handled adding a density estimator to the fitness. This work presents a new strategy, called *Strength by Objective*, which seeks to mitigate said weakness, including within the mechanism those elements from the population that do not dominate nor are dominated by others, which are completely ignored by the *Strength* technique.

Keywords: Multiobjective Optimization, Evolving Algorithms, EMO, SPEA2.

Resumen

En la actualidad, se ha invertido un importante esfuerzo en la determinación de una técnica heurística que encuentre mejores soluciones a optimizaciones multiobjetivo. Entre la gran variedad de propuestas resalta el algoritmo evolutivo SPEA2, por su demostrada superioridad sobre otras técnicas en un surtido grupo de pruebas experimentales. Uno de los pilares de SPEA2 es su estrategia de asignación de *fitness*, llamada *Strength*, que clasifica a la población con un mecanismo de nicho, basados en el concepto de Frente Pareto. Aunque esta estrategia se considera una de las mejores por los resultados obtenidos, ella presenta una debilidad que es manejada agregando un estimador de densidad al *fitness*. El presente trabajo propone una estrategia, denominada *Strength By Objective*, que busca mitigar dicha debilidad, incluyendo dentro del mecanismo a aquellos individuos que no dominan ni son dominados por otros, los cuales son ignorados completamente por *Strength*.

Palabras claves: Optimización Multiobjetivo, Algoritmos Evolutivos, EMO, SPEA2.

1. INTRODUCCIÓN

Cuando un problema matemático consiste en optimizar diversas funciones que comparten el mismo espacio de búsqueda, y entre dichas funciones no se puede establecer ningún tipo de ponderación o prioridad, se dice que es un problema de optimización multiobjetivo. Además, en muchos casos, estas funciones pueden ser contradictorias. Tomemos como ejemplo la función Schaffer F_2, usada en el trabajo de Eckart Zitzler y Lothar Thiele [1] (**Figura 1**), que consiste en conseguir un mismo numero real X, que minimice a g(x) y a h(x). A simple vista, se puede observar que si g(x) tiende a su mínimo global, h(x) se aleja de su mínimo global, y viceversa.

$$F_2(x) \begin{cases} g(x) = x^2 & \rightarrow \text{Mínimo global } g(0) = 0 \\ h(x) = (x-2)^2 & \rightarrow \text{Mínimo global } h(2) = 0 \end{cases}$$

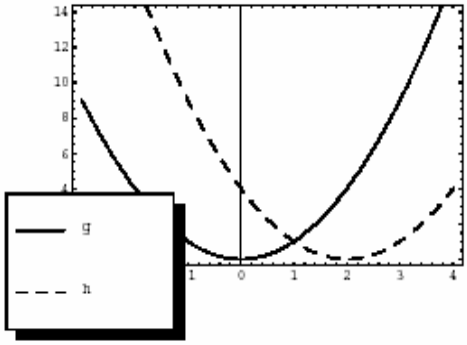


Figura. 1. Schaffer F_2 [1]

Observemos la siguiente comparación:

Para x = 0 → F_2 (0) = (g(0), h(0)) = (0,4)
Para x = 2 → F_2 (2) = (g(2), h(2)) = (4,0)

Si mantenemos el principio de no ponderación de los objetivos, no se puede establecer cual de las dos soluciones es mejor. Por esta razón, la solución a un problema multiobjetivo suele ser un conjunto, al cual se le denomina Frente Pareto. Este conjunto esta formado por aquellas soluciones que no consiguen ninguna otra solución que las mejore en todas las funciones que se busca optimizar. Estas soluciones pareto óptimas también son llamadas soluciones no dominadas. Las soluciones dominadas, son aquellas que si consiguen a otra solución que las mejora en todos los objetivos. En el tutorial escrito por Zitzler, Laumanns, y Bleuler [3] se puede encontrar una buena introducción a la optimización multiobjetivo.

La segunda versión de *Strength Pareto Evolutionary Algorithm* (SPEA2)[2], es una de las técnicas heurísticas mas utilizada en la resolución de optimizaciones multiobjetivo. Su demostrada superioridad sobre otras estrategias, como NSGA-II [4] y su antecesor SPEA [1], lo han colocado como una de las primeras opciones a considerar cuando se presenta un problema de optimización de este tipo.

SPEA2 se diferencia de otros algoritmos evolutivos multiobjetivos por varios factores, entre los cuales tenemos dos de suma importancia ya que permiten determinar la calidad de un individuo:

- Una estrategia de asignación de *fitness*, que toma en cuenta para cada individuo cuantos individuos domina (fuerza) y por cuantos otros es dominado (golpes recibidos). En este trabajo denominaremos *Strength* a esta forma de cálculo de *fitness*.
- Utiliza una técnica de estimación de densidad denominada el *K-ésimo vecino más cercano*, que refina el *fitness* y otorga una guía mas precisa para el proceso de búsqueda.

La estrategia de asignación de *fitness* se divide en dos partes: La primera determina la fuerza de cada individuo, que es igual a la cantidad de individuos que son dominados por él. En la segunda, cada individuo recibirá golpes por parte de todos los individuos que lo dominan. La magnitud de esos golpes vienen definidos por la fuerza del golpeador.

La suma de las magnitudes de los golpes recibidos es el *fitness* del individuo. Nótese que a menos golpes recibidos, el individuo es mejor, lo que indica que la función *fitness* es para minimizar.

Entendiendo bien esta estrategia, se puede observar claramente un detalle. Si la mayoría de los individuos no se dominan entre si, existirán grandes grupos de individuos que poseen el mismo *fitness*, lo cual no permite clasificar correctamente la población y por ende los individuos serán escogidos prácticamente de forma aleatoria. Para mitigar este detalle, SPEA2 agrega al *fitness* el estimador de densidad, el cual permite desempatar a aquellos individuos con el mismo *fitness*.

Pero además, esta estrategia no toma en cuanta a los individuos que no son dominados ni dominan a otros, ya que estos no le dan fuerza a ningún individuo y además no golpean a nadie. Se puede hasta decir, que la existencia de estos individuos no afecta para nada la estrategia de asignación de fitness del resto de la población. Definitivamente, aquí se esta perdiendo información.

2. SPEA2: STRENGTH PARETO EVOLUTIONARY ALGORITHM 2

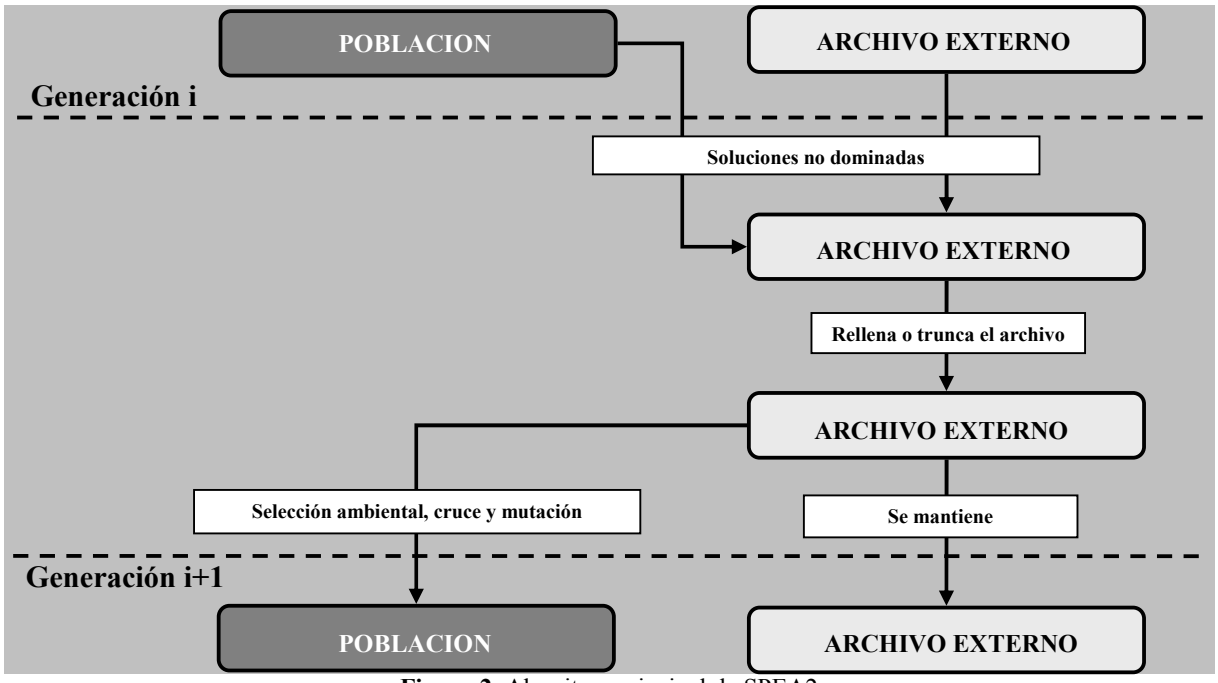


Figura 2. Algoritmo principal de SPEA2

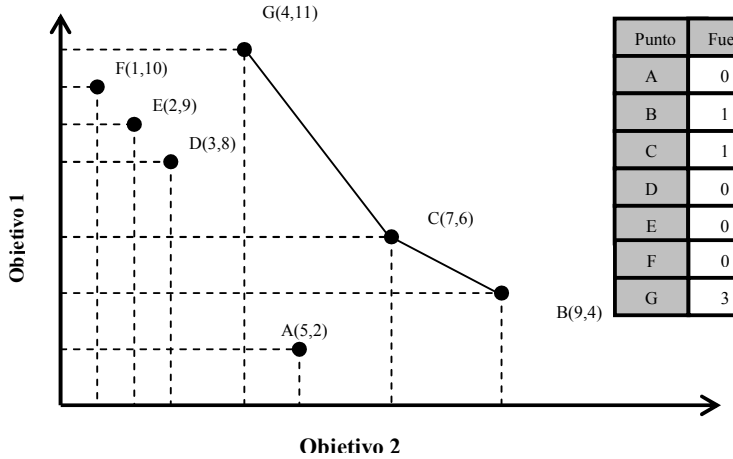
Esta técnica heurística presenta un algoritmo evolutivo sencillo (**Figura 2**) que posee el siguiente hilo de ejecución principal:

1. Calcula la bondad o *fitness* de la **Población** y del **Archivo externo.**
2. Actualiza el **Archivo externo**.
3. Realiza **torneo binario** sólo con los elementos del **Archivo externo**.
4. Cruza a los vencedores del torneo, obteniendo una nueva población.
5. Vuelve al paso 1, hasta que se cumpla la condición de parada.

2.1. La estrategia de asignación de fitness de SPEA2

Este es el punto fuerte de la heurística. SPEA2 ya que presenta un sistema de valoración de soluciones basado en la dominancia entre vectores. Cada individuo posee dos atributos:

a) Fuerza: Representa la cantidad de individuos dominados por él.
b) Bondad: Es la suma de las fuerzas de los individuos que lo dominan.



Punto	Fuerza	Fitness
A	0	2
B	1	0
C	1	0
D	0	3
E	0	3
F	0	3
G	3	0

Figura 3. Asignación de fitness usada por SPEA2

Para hacer el cálculo del fitness de todos los individuos (los de la Población y los del Archivo externo), primero se le asigna a cada individuo su fuerza, determinando a cuántos individuos domina. Si observamos la **Figura 3**, que representa un problema donde se desea maximizar ambos objetivos, podemos decir que los puntos B y C dominan a A, mientras G domina a D, E y F. Luego, a cada individuo se le asigna su *fitness,* sumando las fuerzas de los individuos que lo dominan. Como ejemplo, el *fitness* del punto A es dos (2) por que es dominado por B y C ya que la fuerza de ambos vale uno (1). Como se puede ver también en la **Figura 3**, a los individuos pertenecientes al frente pareto (puntos B, C y G) no los domina nadie, por lo cual, su *fitness* es cero (0).

Pero si los individuos de la población establecen pocas relaciones de dominancia entre si (son todos del frente pareto, por ejemplo), se formarán grandes grupos de individuos con el mismo *fitness*, lo cual hará que el algoritmo escoja a los individuos con un alto grado de aleatoriedad. Para mitigar este detalle, a este *fitness* se le suma el estimador de densidad denominado *K-ésimo vecino más cercano*. Primero, para cada individuo se debe calcular la distancia cartesiana que existe entre él y cada otro individuo de la población y del archivo externo en el espacio solución. Con esto obtenemos un vector de distancias ($dist(x)$), el cual debe estar ordenado de menor a mayor. Luego se toma el *K-ésimo* elemento del vector (donde $k = \sqrt{tamañoDelaPoblacion + tamañoDelArchivo}$).

La formula del estimador de densidad *D* para un individuo *i* queda así:

$$D(i) = \frac{1}{dist(k) + 2}$$

La constante 2 que se suma en el denominador es para garantizar que el estimador de densidad siempre sea menor que cero. Con esto se puede decir que todo individuo pareto óptimo tiene un fitness menor que uno (1).

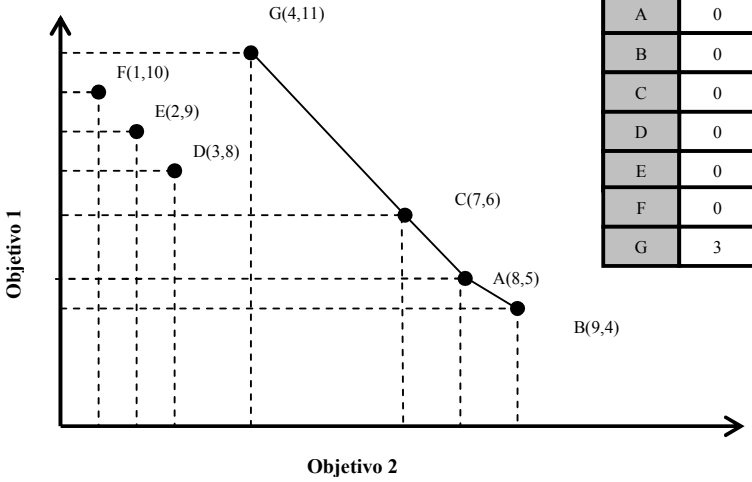
2.2. La actualización del archivo externo

El Archivo externo es una lista de las mejores soluciones que se han conseguido hasta la generación actual. El proceso de actualización de este archivo externo cumple con los siguientes pasos:

- **a)** Se calculan las soluciones no dominadas de la Población actual y del Archivo externo.
- **b)** Se genera un nuevo archivo externo con las soluciones conseguidas en el paso anterior.
- **c)** Si el tamaño del nuevo Archivo Externo sobrepasa un tamaño preestablecido, se utiliza un operador de truncamiento basado en distancia cartesiana de vectores. Si el tamaño del archivo es menor que dicho tamaño, se rellena el Archivo Externo con los mejores individuos de la Población, basándose en la bondad o *fitness* de éstos.

2.3. El torneo binario (con reemplazamiento)

La selección se realiza única y exclusivamente con los individuos que se encuentran en el Archivo externo. Los individuos generados por los cruces son considerados la población para la próxima iteración del algoritmo. El operador de cruce usado es el *single point crossover* con punto aleatorio.



Punto	Fuerza	Fitness
A	0	0
B	0	0
C	0	0
D	0	3
E	0	3
F	0	3
G	3	0

Figura. 4. Un detalle en la asignación de fitness de SPEA2

3. STRENGTH BY OBJECTIVE

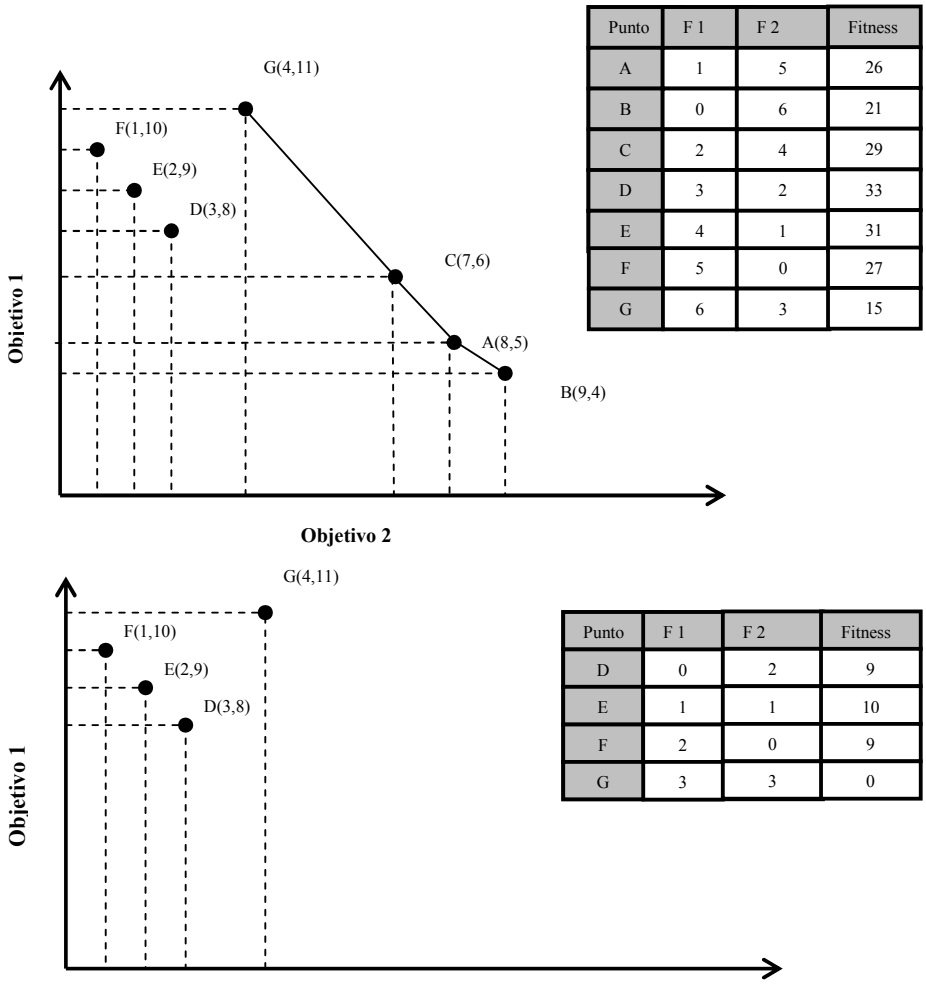
En el ejemplo de la **Figura 4,** donde se desea maximizar dos objetivos, se puede observar como los individuos A, B y C no dominan ni son dominados por nadie. Por ende su fuerza y sus *fitness* son cero (0). Pero el resto de la población si establece alguna relación de dominancia con al menos un individuo de la población.

Cuando se calcula el *fitness* de todos los individuos podemos observar que los individuos A, B y C son ignorados completamente por el algoritmo. Tanto así que si eliminàramos esos puntos y recalculáramos el *fitness* de la población restante, esta no cambiaria. Esto lleva a pensar que en el algoritmo existe una cierta perdida de información, ya que existen individuos que son incapaces de informar al resto de la población de su existencia.

Strength by Objective es nueva estrategia de asignación de *fitness* busca minimizar las debilidades presentadas por *Strength*, enfocándose estos dos problemas:

- Grandes grupos de individuo con el mismo fitness.
- Individuos que son ignorados por el algoritmo.

Esta estrategia de asignación de *fitness* aborda el problema de forma más granular, tomando en cuenta la calidad de individuos en cada uno de las funciones a optimizar.



Punto	F 1	F 2	Fitness
A	1	5	26
B	0	6	21
C	2	4	29
D	3	2	33
E	4	1	31
F	5	0	27
G	6	3	15

Punto	F 1	F 2	Fitness
D	0	2	9
E	1	1	10
F	2	0	9
G	3	3	0

Figura 5. Strength By Objective

Strength by Objective utiliza atributos equivalentes a los de SPEA2 para cada individuo, aunque se calculan de forma distinta:

1. *Fuerza por objetivo*: Este atributo indica a cuántos individuos de la población vence el individuo en un objetivo en particular. Nótese que se tiene una fuerza por cada objetivo a optimizar.
2. *Golpes recibidos*: Cada vez que un individuo es vencido por otro en un objetivo recibe un golpe. La magnitud de este golpe depende de la fuerza del vencedor para ese objetivo. Este atributo es la suma de los golpes recibidos por un individuo.

```
Método calculo de Fuerzas (Individuo  A)
  Para cada individuo B de la población y el archivo

    Para cada objetivo i a optimizar
      Si (A.vectorSolucion[i] es mejor que B.vectorSolucion[i])
        A.fuerza[i] ++;
```

Figura 6. Algoritmo para el cálculo de Fuerzas

Primero se calculan las fuerzas de los individuos (**Figura 6**). Para esto, se compara a cada uno de ellos con el resto de la población y se determina a cuántos individuos vence en cada objetivo a optimizar. Luego, se determinan cuántos golpes recibe un individuo comparando a cada individuo con el resto de la población (**Figura 7**). En cada comparación, si el individuo vence a otro de la población en un objetivo, lo golpea con la fuerza que tenga para ese objetivo. La bondad o *fitness* de un individuo es la cantidad de golpes recibidos. A menor *fitness*, el individuo es mejor.

```
Método Dar Golpes (Individuo  A)
  Para cada Individuo B de la población y el archivo
    Para cada objetivo i a optimizar
      Si (A.vectorSolucion[i] es mejor que B.vectorSolucion[i])
            B.golpesRecibidos = B.golpesRecibidos + A.fuerza[i];
```

Figura 7. Algoritmo de asignación de *fitness*

Nótese que con esta estrategia la población completa participa en el proceso, porque todo individuo cumple con la condición de perder o ganar en un objetivo determinado. Además, este enfoque disminuye las posibilidades de que grandes grupos individuos tengan el mismo *fitness*.

Otro detalle que se debe resaltar acerca de *Strength By Objective* es que determina un nuevo orden a la población. Si se observa a los individuos A, B, C y G de la **Figura 5,** que representa un problema de maximización de dos objetivos, notamos que son del frente pareto, pero sus *fitness* son completamente distintos, sin necesidad de agregar alguna otra estrategia, como el estimador de densidad.

5. STRENGTH VS STRENGTH BY OBJECTIVE

La forma de determinar si la nueva propuesta aborda correctamente el problema, es sometiéndolo a una comparación experimental. Como punto de comparación se va a utilizar la estrategia de asignación de fitness *Strength*, de SPEA2.

5.1. El algoritmo

Se hizo una implementación de SPEA2 siguiendo las especificaciones del trabajo de Zitler, Leummans y Thile [2], que pueda utilizar *Strength* o *Strength by Objective* como estrategia de asignación de *fitness*. Es importante resaltar que

ambas estrategias no usaran el estimador de densidad para así realizar una comparación exclusiva entre las estrategias de cálculo d*e fitness*.

5.2. Descripción de la prueba

Se usaron ambas estrategias para resolver tres planteamientos del "problemas de la mochila" [1] (problema de combinatoria) con 750 elementos cuyos valores y pesos son generados de forma aleatoria. Los individuos son representados con un cromosoma binario. Los parámetros están especificados en la **Tabla 1.**

	Dos(2) Knapsacks	*Tres(3) Knapsacks*	*Cuatro(4) Knapsacks*
Tamaño población	250	300	400
Tamaño archivo	250	300	400
Mutación	0.006	0.006	0.006
Generaciones	50	50	50

Tabla 1. Parámetros de las pruebas a realizar (Knapsacks problems [1])

5.3. Variables de medición

- *Área cubierta*: Este criterio indica la cantidad de área del espacio de solución cubierta por el frente pareto dado. En un espacio de dos dimensiones, el área cubierta será igual al área de la unión de los rectángulos formados por cada uno de los vectores del frente pareto y el origen (0,0). Este criterio puede ser canónicamente extendido a N dimensiones. La **Figura 8** muestra un ejemplo.
- *Porcentaje de Dominancia*: Dado dos conjuntos frente pareto A y B, el porcentaje de dominancia de A/B indica el porcentaje de soluciones de B que son dominadas por al menos una de las soluciones de A.

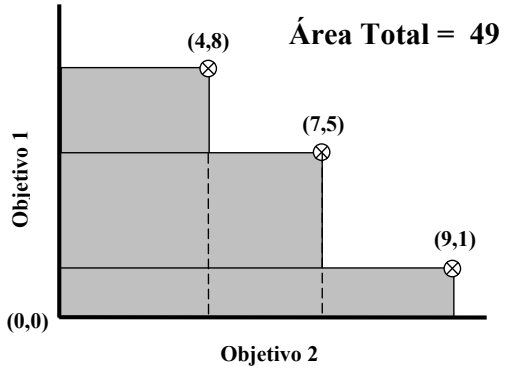


Figura 8. Área cubierta

5.4. Resultados

Estos resultados son el promedio obtenido de diez (10) corridas independientes del algoritmo para cada planteamiento.

- *Área cubierta*

Estrategia	*Dos(2) Knapsacks*	*Tres(3) Knapsacks*	*Cuatro(4) Knapsacks*
Strength	**6,62E+10**	**1,42E+16**	**3,04E+21**
Strength by Objective	6,58E+10	1,41E+16	2,93E+21
Diferencia	4,55E+08	1,25E+14	1,06E+20

Tabla 2. Resultados promedios de área cubierta

- *Dominancia*

Estrategia/Estrategia	*Dos(2) Knapsacks*	*Tres(3) Knapsacks*	*Cuatro(4) Knapsacks*
Strength / Strength by Objective	**45,38%**	28,25%	21,50%
Strength by objective / Strength	44,50%	**29,38%**	**62,50%**
Diferencia	0,88%	1,13%	41,00%

Tabla 3. Resultados promedios de dominancia

5.5. Análisis

En el promedio de área cubierta (**Tabla 2**) se observa que *Strength* se mantuvo por arriba de *Strength by Objective* por una mínima diferencia. Con esto se puede decir que ambas estrategias se comportan igual. Pero la dominancia nos revela que si existe una diferencia. En la **Tabla 3,** se puede observar como *Strength by Objective,* comienza a ganarle terreno a *Strength* al aumentar el numero de objetivos, llegando al punto de sacar una diferencia de 41 puntos porcentuales para el problema de cuatro mochilas (*Four Knapsacks*).

Los resultados obtenidos en la prueba experimental ponen a *Strength by Objective* en muy buena posición. Usando las dos variables de medición, se puede observar que *Strength by Objective* parece mejorar su desempeño con respecto a *Strength* al aumentar el numero de objetivos.

6. CONCLUSIÓN

Strength By Objective es una estrategia de asignación de *fitness* para algoritmos multiobjetivos que evalúa a la población tomando en cuenta la calidad de un individuo para cada función a optimizar. Basándose en *Strength* de SPEA2, esta nueva propuesta establece una nueva clasificación a las poblaciones, la cual busca que todos los individuos de la población colaboren con su evaluación, tratando así de no perder la valiosa información que representa la existencia de un individuo. Pero además, *Strength by Objective* garantiza que la información se propague, ya que un individuo influenciara a todos los demás, porque siempre establecerán una relación de ganar o perder en un determinado objetivo.

Las comparaciones experimentales entre *Strength* de SPEA2 y *Strength by Objective* permiten decir que la nueva propuesta mejora a la de SPEA2 al aumentar la cantidad de funciones a optimizar, y en el peor de los casos ambas estrategias se comportan de la misma forma.

7. EL FUTURO DE LA INVESTIGACION

Este trabajo es solo el comienzo de una serie de investigaciones que buscan determinar el desempeño de *Strength by Objective* en la rama de los EMOs. (Evolutionary Multiobjective Optimization). En los siguientes pasos, se utilizara esta nueva propuesta para la resolución de un amplio grupo de problemas, de espacios de búsqueda continuos y discretos. Otro experimento que ayudara a definir la calidad de *Strength by Objective* es compararlo con el estimador de densidad usado por SPEA2 para corregir su debilidad.

Referencias

[1] Eckart Zitzler and Lothar Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. May 1998.

[2] Eckart Zitzler, Marco Laumanns, y Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. May 2001(Errata added Sept 27, 2001).

[3] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler A Tutorial on Evolutionary Multiobjective Optimization.

[4] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II.2001.

An Architectural Model to Support Collaborative Work through Mobile Devices

César A. Collazos
Department of Systems, FIET, Universidad del Cauca
Popayán, Colombia
ccollazo@unicauca.edu.co

Rosa Alarcón
Department of Computer Science, Pontificia Universidad Católica de Chile
Santiago, Chile
ralarcon@ing.puc.cl

Luis A. Guerrero
Department of Computer Science, Universidad de Chile
Santiago, Chile
luguerre@dcc.uchile.cl

Abstract

Groupware applications play a major role in supporting distributed teams for working together collaboratively. Several groupware platforms have been developed to simplify the development of synchronous and asynchronous applications focusing on relieve developers from facing standard problems such as communication, synchronization, coordination or concurrency control. Upcoming communication technologies (e.g. wireless area networks, cellular telephones, PDAs, pagers, etc.) are a promising and interesting market for developing group support features. However research in groupware systems does not address the need to support individual tasks, which are part of group activities. As development and research in both areas grows and intermingle, we believe that this issue will acquire more relevance. In this paper, we propose an architecture that allows users to move seamlessly across four different scenarios: individual and collaborative work supported by portable devices and desktop computers.

Keywords: Mobile Work, Collaborative Activities.

1. INTRODUCTION

In Computer Supported Cooperative Work (CSCW) area it is widely recognized the benefits that collaborative interaction has on work groups and organizations. Collaboration is seen as "a principle-based process of working together that produces trust, integrity and break-through results by building true consensus, ownership and alignment" [17].

Current problems and modern work require different expertise, due to their complexity and size. Schrage defines collaboration as the "process of shared creation: two or more individuals with complementary skills interacting to create a shared understanding" [22]. That is, as project size and complexity grows, workers face stressed cognitive demands to be efficient and react accurately and then, collaboration and networking can take advantage of distributed expertise and become a key factor for successful problem-solving activities [8].

Groupware systems are computer-based applications that support groups of people working together toward a common goal, by providing them a shared interface to their shared environment [7]. A typical example is a shared editor (also known as "collaborative editor" or "collaborative writing tool"), where "the writing of a document is performed for more than one author" [23].

On the other hand, information technology (IT) devices are becoming increasingly portable, powerful and affordable. The use of computer technologies is no longer confined to the desktop and office settings; users now have access to highly portable personal computing appliances, such as palmtop computers or Personal Digital Assistants (PDAs), which can be used "anytime, anywhere". Such devices have different functions; they can be used to record data, to access information resources or to communicate with other people. It has been argued, therefore, that mobile computing devices could be useful tools for supporting learning and workplace activities [9], either alone or in combination with desktop computers. Handheld computers are emerging as a flexible and portable solution that provides users with "ready to hand" support to engage in collaborative activities anytime, anywhere [16].

However, when we observe collaborative work, we may notice that there are some tasks or activities that need to be performed individually. For instance, when a group works together in a shared editor, users go through phases of self-organization (i.e. organizing their ideas, thoughts and critics) and then, once they have a clear picture of their contribution they go to a sharing phase where they share their thoughts, ideas or critics and can argue and negotiate. The central concern of this paper is that regardless exiting several successful applications that exploit PDAs in individual and shared environments, research in the area of collaborative systems does not address the need for supporting individual tasks, even when handheld devices (originally designed to support individuals) are used. Furthermore, we believe that as mobile devices become a promising technology for supporting work group, the need of supporting the seamless transition between individual and workgroup scenarios would increase.

In this paper, we present an architectural model and a Web-based application that supports the collaborative edition of a text document based on the model proposed. This tool has a handheld version. Section 2 presents some related work. In section 3 and 4 we describe the architecture proposed and the example application. Section 5 presents some benefits of or proposal and finally, in section 6 we present some conclusions.

2. Related Work

Most work on handheld devices for supporting individuals has focused on how can be used to replace stationary computers and how to take advantage of their affordances and mobility. Additionally, some researchers are focused on using handheld devices in group settings to support collaborative work (see [2, 19, 24]). Other approaches are listed below:

1. Mobile devices can serve as a means for people to augment their real time personal communication.
2. Mobile devices allow people downloading common information (such as information stored in databases), modify it, and upload it again.
3. Mobile devices allow people gathering personal information in the field, which can be downloaded into a common database, for example, in NotePals, participants at a conference session write meeting notes on their personal PDAs [5].
4. Some mobile systems synchronize and negotiate personal information across devices (i.e. 3Com Palm Pilot Hotsync Manager replicates information held in equivalent applications across both PDA and desktop computers [1]).
5. Synchronization systems are powerful tools that allow people synchronizing both their personal and public information across devices. Using Lotus EasySync, for example, information within Palm Pilot applications are synchronized with a Lotus notes database [15].
6. Another approach is to give mobile person access to one's workstation environment. For instance, PalmVC allows the mobile PDA user connecting to and browsing a portion of his workstation screen [18].

Mobile computing encompasses the situation where people produce and share information in a mobile work setting. In the practice, however, the majority of mobile computing systems neglect the nuances between personal and public information exchange [10]. Our specific research interest is trying to understand the nuances of how people moves across personal and public information (in our case, within group boundaries) to share information or common data.

3. System Architecture

Designing PDAs-based applications requires to take into account both restrictions and opportunities. We consider as restrictions: PDAs' reduced screen size, limited available memory, heterogeneity [3], input facilities and processor

speed, small communications bandwidth and when wireless support is used, eventual interruptions in information interchange as a consequence of wireless communication intermittence [13]; if there is no support for wireless communication, interruption can be longer, and then reconnection and consistency management is required (synchronization tasks). Therefore, it is important to consider the following design issues:

- Design simple user interfaces that include few elements.
- Consider limited memory and storage. Most data stored in the PDA is volatile.
- Consider alternative data input devices. PDAs are slow and their design is not intended for supporting the input of large amounts of data.

Opportunities provided by handhelds include: great portability, short start-up and response times and ability for gathering and presenting small pieces of information. Thus, the following design characteristics are suggested by these opportunities:

- PDAs may be useful when user mobility is a main consideration during the operation of the system.
- Using PDAs when individual work is required in unusual, congested or uncomfortable places.
- Consider the use of PDAs when users can make timely short annotations that can be expanded into larger contributions afterwards.
- Consider the use of PDAs during periods of individual divergent work. These periods will probably be followed by a period of group convergent work, where individual production is appropriately merged and improved.

The role of individuals in collaborative work should not be overlooked. An interesting study exploring the opposing individual and collective views in the foundations of group work in organizations has been done by DeSanctis [6]. He argues that elements in existing theory related to IT-based group support systems must include: the recognition of the importance of socio-emotive communication in group functioning; the influence of group norms and values in information processing by groups; the diverse nature of group membership; and the issue of individual freedom versus managerial control and power. The people collaboration degree, and the degree to which tasks require collaboration, vary substantially, not only between jobs but also within jobs over time. Many jobs, tasks or activities can be thought of as moving continuously from being performed individually to being performed collaboratively. With those concerns, we have designed an application that allows a seamless transition between individual and shared activities. Figure 1 depicts the proposed system architecture. It is composed by four layers: (i) presentation, (ii) logic, (iii) synchronization and (iv) data.

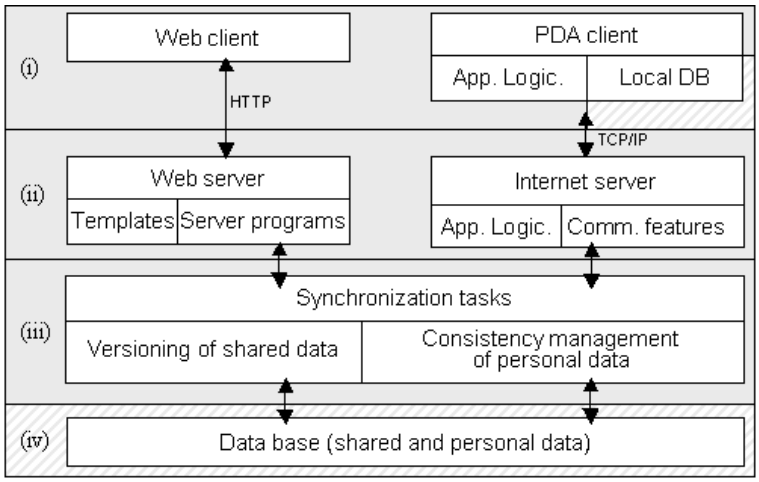


Fig. 1. System Architecture

In the presentation level, a Web client provides an interface to a Web version, which is a standard shared editor, while the PDA client allows users to edit documents on a PDA. In this case, a local copy is stored in the local database and application's logic is also included in order to provide "wireless unsupported" mode.

In the logic layer it is also implemented communication and security features. Web server answers Web client requests for transactions on the database. It has two sub-layers one supporting presentation tasks through Templates adapted to users' roles (i.e. reading through, making comments, etc.) and programs running on the Web Server that implement the logic itself. The Internet server is a network host that handles PDA client communications' tasks (i.e. yellow pages). In the synchronization level the code supporting the versioning of documents created in group or "collaborative" mode is implemented. Users manipulate those versions from the Web or the PDA client. Individual work is supported trough consistency checking routines, that is, users can create personal notes that are treated separately and integrated in shared versions when users demand it. Finally the data layer contains a database that comprehends group versions as well as personal notes.

4. A SAMPLE APPLICATION

Based on the architecture presented in the previosu section we have implemented a software tool named Ecomov. This tool is a collaborative writing tool composed of four modules: a user management component, a Web editing module, a PDA text-editing module, and a communication and synchronization component.

Work supported by PDAs may be done in two ways: network-connected (on-line) and off-line. When working on-line, document synchronization is automatic, while off-line PDA work occurs when the co-author steps outside the range of the wireless network. In the latter case, the PDA stores a "copy" of the original document and enables co-authors to perform text editing as desired. Of course, after off-line work, the performed changes are synchronized with the master document and a new version (of the master document) is stored. This implies that all stored versions must be merged (synchronized) from time to time. A coordinator does this merging and usually this involves discussion with the other co-authors in order to keep document coherence. This task must be done with the agreement of the other co-authors (they are aware of the occurrence of this task), so that, previous coordination tasks must be performed.

4.1. WEB EDITING MODULE

This module allows users to create, edit and share documents through the Web. When a group member creates a new document, s/he must provide the list of co-authors and their roles (the current version just considers reader and reader/writer roles). When a co-author modifies a document, it is blocked and after making changes, s/he must unblock it. This module also allows co-authors to generate a new document version. Finally, co-authors can add their own personal annotations and see annotations provided by other users. Figure 2 shows a document being edited via Web.

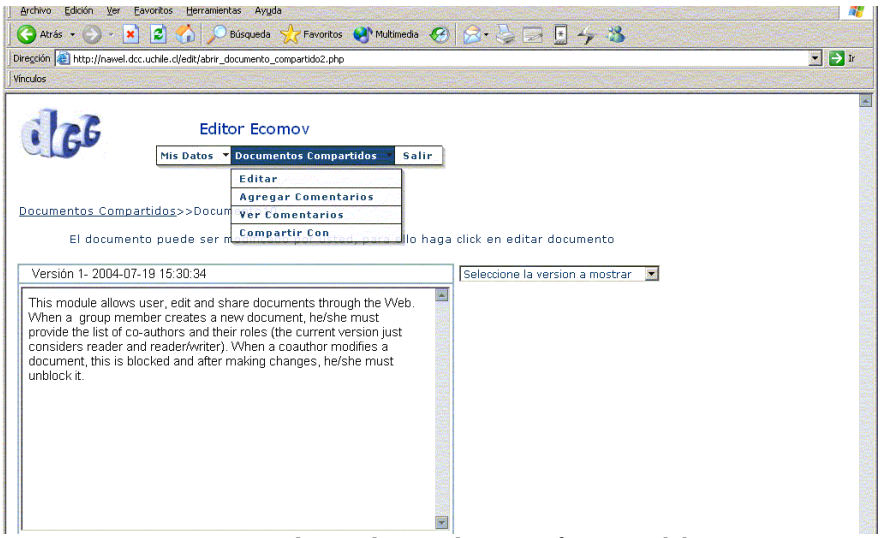


Fig. 2. Ecomov editor: Editing a document from a Web browser

4.2. PDA EDITING MODULE

In the design, we tried to make the Web and PDA editing modules as similar as possible. However, considering the strategy outlined in Sect. 4, the PDA module should be very compact, with a simple user interface, and using little storage and at the same time, it should still provide the main functionality of the Web module.

Co-authors may create new documents or open previous ones. These may be personal or shared. A typical use may be to create a personal document with an outline of ideas; these are expanded later in a shared document. Shared documents may have several versions, which can be browsed by the co-author. The user can also place annotations on any document from this software module.

Figures 3a and 3b show the PDA editing module user interface. Users can open shared or personal files (File options), in both cases an interface presenting a list of the available documents is shown. For the case of shared files, the status (modified or not) as well as the co-author name that made the last modification is also presented (Fig. 3a). Once the user pick-ups a document, it is presented for editing. In this case additional information such as the document version is also presented (upper part of Fig. 3b). The middle part of the screen shows the document, and the lower part contains the application menu. Fig. 3b shows the "File" menu options. The file menu allows users to save the current version or generate a new one, as well as to synchronize local versions with the central server. The editing menu has an option to work on various versions: the buttons labelled "<<" and ">>" allow users to move forward or backwards on the local document versions.

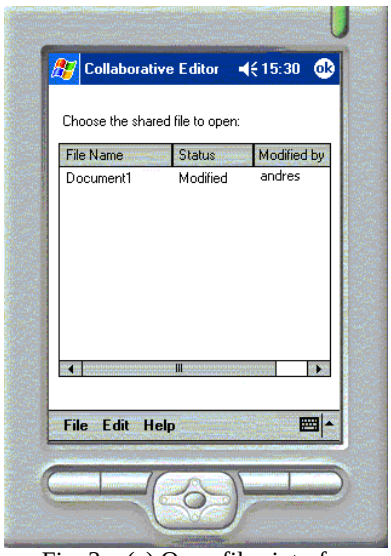


Fig. 3. (a) Open files interface

(b) File menu options and editing interface

4.3. COMMUNICATION AND SYNCHRONIZATION MODULE

This module allows communication and synchronization between PDAs and the server database. Users must choose the local version they want to synchronize with the server database. As the Web editing modules also accesses the central database, the main difficulties solved by the communication module concerns concurrency and consistency, since several co-authors could be editing the same document at the same time. The module also solves the document versions management problem. Keys for a simple solution to these problems are the locking mechanism already mentioned and a time stamp associated by the system to each document version. Time stamps are then used by the system itself to guide co-authors on which versions are appropriate for merging.

The locking mechanism is paired with a unique version of the document, called the master document, as introduced above. When a co-author has blocked the master document, other co-authors can make annotations over copies of the master document. If different changes have been made to the document, they are manages as if they were annotations: at a later time, a co-author can modify the master document based on all her colleagues' annotations. For such task, the

system lets visualize all document copies associated to a master document as separate windows. Annotations are shown in color to make them easily distinguishable.

5. DISCUSSION

One of the main advantages of handheld computers is that they are portable. Also, initial vendor-supplied software encouraged individual rather than collaborative use. Even now, most handheld applications often reinforce the idea of a handheld computer as a personal digital assistant. Nevertheless, recent articles describe group or collaborative applications based on this kind of devices (see [14]). In particular, some developers may be interested on designing systems to support people doing cooperative tasks. However, a few relevant questions should be asked: are handhelds appropriate components for collaborative applications? If so, which are the tasks they support the best? There are several mobile applications that successfully support individuals but not in a collaborative or work group setting. On the other hand, mobile applications that support collaborative activities are mainly focused on the group component of activities, neglecting the individual work component. So, does it make sense to treat them as separated development environments, leaving users the responsibility of making the necessary compatibility changes? Or, due to their nature, are they separated environments?

Sharing and managing expertise appear, almost by definition, to be activities that require collaboration between people, consequently, we envision and design systems for expertise management that assume collaboration as the starting point. Unfortunately, this view is naive, because it does not recognize that individuals' concern is more than likely, developing, using and creating their own expertise. The need to share expertise takes place later, perhaps when others demand it, or by serendipity, or perhaps as a side effect of a group's need to coordinate and share their knowledge in order to develop a project or solve a hard problem.

Everyday working styles of people, including the way of sharing expertise, shift regularly and easily between individual and group activity. This implies that software should support both individual and group needs. Unfortunately, most systems are categorized either as a single-user application or as multi-user groupware and the gap created for people moving between individual and group activities force them switching tools, causing frustration and additional work on them.

Single-user applications offer little or no explicit support for people who want to work together through computers. While people do often use single user systems collaboratively [20], the effort to do so across time and distance barriers is high. Nardi's 1991 study of spreadsheet users, for example, clearly describes how co-located people share expertise over spreadsheets: one person developing the content of a spreadsheet is helped by another who understands how to program spreadsheets; this style of interaction is far more difficult when people are geographically distributed. On the other hand, groupware built to fit only group needs is often inadequate for supporting individual work. The gap between these categories often means that people cannot use the same tools for tasks that are conceptually similar but cross over individual and group work [12]. People must take on the additional burden of shifting between tools, and learning how to use new tools. They must translate common artifacts, such as documents, into formats amenable to both single and groupware systems. Cockburn and Thimbleby suggest that CSCW environments must not only support group activity, but must also cater to individual requirements [4].

The transition between individual and collaborative work could be very fluid, in the same way that transition between task-oriented and non task-oriented activities or between several tasks [21]. Various in-depth studies have shown how apparently separated working people collaborate now and then in a very subtle way [11]. So, although collaborative work generally refers to situations where two or more people act together explicitly, to achieve a common goal, the actual extent of "togetherness" can vary substantially. Designers of collaboration technology should therefore take account of the fact that all collaborative tasks are a combination of individual and collaborative activities. Group work is also individual work. Group work is a mixture of collaborative activities and individual or subgroup task performance.

6. CONCLUSIONS

Collaborative applications significantly differ from single-user applications. Many users provide input (often simultaneously), output has to be processed for many users and shared data have to be kept consistent. On the other

hand, when we observe a collaborative work activity we may notice there are some tasks, which are performed individually in many cases. Workers are autonomous and tend to distribute their workload, so their need for ongoing coordination and planning is minimized. This implies that work division is mutually understood within the group so that workers do not need to check constantly with each other. Of course, it is not obvious if handhelds may be adequate for ambitious systems involving several people with many complex interactions among them.

There are several problems that must be solved in collaborative environments supported by mobile devices. Collaborative and mobile applications have to keep data consistent. Applications which are collaborative and mobile at the same time double the problem of data consistency. Collaborative applications have to synchronize concurrent data manipulations, mobile applications have to keep data consistent when devices are moved inside the network or are disconnected from the network.

The model proposed in this paper could be useful to design groupware application taking into account the transition between individual and shared activities. However, what is clear is that mobile devices such as PDAs can support efficiently individual tasks that are developed in a collaborative context and implies users' mobility mainly in environments where there are no network supports (off-line). For instance, reading a document while you are in the subway. When these applications are used in more complex environments like large organizations, we must recognize that beyond personal support, users are involved in work group and move constantly from an isolated working mode to a sharing phase where they coordinate their actions and collaborate, and then, a seamless transition between those phases could ease their work and lessen their frustration due to additional workload.

ACKNOWLEDGMENTS

This work was partially supported by Fondecyt (Chile) grant No. 1030959.

REFERENCES

[1] 3Com Corporation: Palm Pilot Handbook.
[2] Abowd, G., Investigating the capture, integration and access problem of ubiquitous computing in an educational setting. Proceedings of SIGCHI'98: Human Factors in Computing systems (1998), pp. 440-447.
[3] Cannataro, M., and Pugliese, A. A survey of architectures for adaptive hypermedia. In Levene, M. and Poulovassilis, A. (Eds.): Web Dynamics, Springer Verlag (2004), pp.357-386.
[4] Cockburn A., and Thimbleby H. A reflexive perspective of CSCW. ACM SIGCHI Bulletin (1991) 23(3), pp. 63-68.
[5] Davis, R., Lin, J., Brotherton, J., Landay, J., Price, M. and Schilit, B. A framework for sharing handwritten notes. Proc. of the ACM Symposium on User Interface Software and Technology, ACM Press, (1998) pp.119-120.
[6] DeSanctis, G. Shifting Foundations in Group Support System Research. In Group Support Systems – New Perspectives, L. Jessup, J. Valacich (eds.), MacMillan Pub. Co., New York (1993), pp. 97-111.
[7] Ellis, C., Gibbs, S., and Rein, G. Groupware, Some Issues and Experiences. Communications of the ACM (1991) 34(1), pp. 38-58.
[8] Engestrom, Y. Developmental studies of work as a test bench for activity theory. Chaik-lin, S. and Lave, J.: Understanding practice. NY: Cambridge University Press (1997).
[9] Fung, P., Hennessy, S. and O'Shea, T. Pocketbook Computing: A paradigm Shift? Computers in the Schools (1998) 14(3/4), pp.109-118.
[10] Greenberg, S., Boyle, M. and LaBerge, J. PDAs and shared public displays: Making personal information public and public information personal. Personal Technologies, (1999) 3(1), Elsevier.
[11] Heath, C., and Lupp, P. Collaboration and control; crisis management and multimedia technology in London Underground line control rooms. CSCW (1992) 1, pp. 69-94.
[12] Ishii H., Kobayashi M., & Grudin J., Integration of interpersonal space and shared workspace: Clearboard design and experiments. ACM Transactions on Information Systems, October, (1993) pp. 96-125.
[13] Joseph A. D., Tauber J. A., and Kaashoek M. F., Mobile Computing with the Rover Toolkit, IEEE Transactions on Computers (1997) 46(3), pp. 337-352.
[14] Kirda, E., Fenkam, P., Reif, G., and Gall, H. A service architecture for mobile team-work. Proceedings of 14th International Conference on Software Engineering and Knowledge. Eng. Ischia, Italy (2002).
[15] Lotus Notes Inc.: Lotus EasySync Version 2.1. Lotus Development Corporation, Cambridge, Ma, USA, (1998).
[16] Luchini, K, Quintana, C., Curtis, M., Murphy, R., Krajcik, J., Soloway, E., and Suthers, D. Using Handhelds to support collaborative learning. Proceedings of CSCL (2002), pp.704-706.

[17] Marshall, E., The collaborative workplace. Management Review (1995) 24, pp.13-17.
[18] Minenko, V. Palm VNC: virtual Network Computing Client for Palm platform http://www.icsi.berkeley.edu/~minenko/PalmVNC (1998).
[19] Myers, B., Stiel, H., and Gargiulo, R. Collaboration using multiple PDAs connected to a PC. Proceedings of the ACM CSCW'98 (1998) pp. 285-294.
[20] Nardi, B. Twinkling lights and nested loops: Distributed problem solving and spreadsheet development. In Greenberg, S. (ed) Computer Supported Cooperative Work and Groupware. Academic Press, London (1991). pp. 29-52.
[21] Robinson, M. Design for unanticipated use. In G. DeMichelis, C. Simone and K. Schmidt (Eds.), ECSCW'93. Proceedings of the Third European Conference on Computer Supported Cooperative Work. Dordrecht: Kluwer Academic Publisher, (1993).
[22] Schrage, M. Shared Minds. New York: Random House, (1990).
[23] Speck, W., Johnson, T., Dice, C. and Heaton, L. Collaborative Writing: An Annotated Bibliography. Greenwood Press, London, UK, (1999).
[24] Want, R. An overview of the ParcTab Ubiquitous computing experiment. IEEE Personal Communications (1995), pp. 28-43.

Ambiente Computacional para la Edición Cooperativa de Diagramas de Clases con UML

Carlos A. Cobos
Universidad del Cauca, Departamento de Sistemas,
Popayán, Colombia, 572
ccobos@unicauca.edu.co

Jaime A. Salazar
Universidad del Cauca, Programa de Ingeniería de Sistemas,
Popayán, Colombia, 572
jsalazarcabrera@ucauca.edu.co

y

Leydy C. Muñoz
Universidad del Cauca, Programa de Ingeniería de Sistemas,
Popayán, Colombia, 572
cmunoz@ucauca.edu.co

Abstract

This paper shows a technique or method of cooperative learning called “Cooperar” and the design of an computational environment for the cooperative edition of class diagrams with UML called "SCU" that will be used for the development of activities (exercises / workshops / evaluations) in the Object Oriented Programming courses of the Systems Engineering Program at the University of Cauca, Colombia, and that it can be useful for other University contexts. The computational environment is designed to take a group of statistics allows to each student to obtain a constant feedback at the same time serves as an indicator of his participation in the design of the diagram, as well as, a service that allows the reproduction of the actions that all the students carried out in each group like a "video". In this way, it is possible diminishing the work load during the process of monitoring and evaluation of the cooperative activity. This reproduction service is based on an adaptation of the concept of the Transaction Log of databases management servers and is free of the problems and storage excess that have outlined other research projects on the topic.

Keywords: Collaborative Systems, Computer Support for Cooperative Learning, Object Oriented Programming, UML, Class Diagrams.

Resumen

Este artículo describe una técnica o método de aprendizaje cooperativo denominado “Cooperar” y el diseño de un ambiente computacional para la edición cooperativa de diagramas de clase con UML denominada “SCU”, que se utilizará para el desarrollo de actividades (ejercicios / talleres / evaluaciones) en los cursos de Programación Orientada a Objetos del Programa de Ingeniería de Sistemas de la Universidad del Cauca, Colombia, y que puede ser útil para otros contextos Universitarios. El ambiente está diseñado para llevar un conjunto de estadísticas que le permiten obtener una realimentación constante al estudiante, que a la vez sirve como un indicador de su participación en el diseño del diagrama, y un servicio que permite la reproducción de las acciones que realizaron todos los estudiantes en cada grupo como un “video”; disminuyendo de esta forma la carga del docente en el proceso de monitoreo y evaluación de la actividad cooperativa. Este servicio de reproducción se basa en una adaptación del concepto del Registro de Transacciones de un motor de bases de datos y está libre de los problemas de rendimiento y exceso de almacenamiento que han planteado otras investigaciones sobre el tema.

Palabras claves: Sistemas Colaborativos, Aprendizaje Cooperativo Soportado por Computador, Programación Orientada a Objetos, UML, Diagramas de Clase.

1. INTRODUCCIÓN

En los últimos cuatro (4) años, los docentes de la asignatura de Programación Orientada a Objetos (POO) de la Universidad del Cauca - Colombia han observado que a los estudiantes se les dificulta aprender los conceptos relacionados con clases y objetos, el manejo de la complejidad, los tipos abstractos de datos (TAD's), el manejo dinámico de objetos, la sobrecarga de operadores y las plantillas (templates). Lo anterior se ha convertido en una de las causas preponderantes para que en esta asignatura el índice de mortalidad académica sea muy elevado. En promedio el 60% de los estudiantes que inician el curso, no lo aprueban o se retiran.

En junio de 2004 se realizó una encuesta [1] a docentes que impartieron e impartían la asignatura de POO y a estudiantes que la cursaron y la estaban cursando en ese momento. En los resultados obtenidos se concluye entre otras cosas lo siguiente: A los estudiantes les hace falta mayor motivación por aprender los conceptos de esta asignatura; esto conduce a que el estudiante se preocupe principalmente en aprobar la materia o los exámenes con la mínima nota requerida que corresponde a 3.0 sobre 5.0, lo cual podría sugerir que el estudiante está adquiriendo alrededor del 60% de los conocimientos y habilidades del curso. Los estudiantes no usan una técnica adecuada de estudio, las técnicas utilizadas dan mayor importancia a la memorización que a la comprensión, ejercitación y aplicación de los conocimientos. Los estudiantes participan muy poco en clase, debido principalmente al temor que sienten por realizar preguntas que carezcan de importancia o por el temor al señalamiento o cuestionamiento por parte de sus compañeros. Esta situación puede llevar a que los estudiantes no comprendan en su totalidad los temas vistos en clase. Finalmente, muchos estudiantes no participan porque se sienten rezagados ante el alto grado de participación y competitividad que muestran algunos estudiantes que poseen un mayor conocimiento en el tema que se trata en una determinada clase.

También se encontraron algunas fallas (o situaciones a mejorar) por parte de los docentes que orientan estos cursos, ellas son: La presentación de los contenidos en algunos casos se hace desorganizadamente, con repetición de contenidos y en algunos casos sin relacionar los conceptos; actualmente, cada docente por separado trata de mejorar los contenidos, pero estos esfuerzos no se realizan coordinadamente. Existe una notable falta de tiempo para el desarrollo de prácticas, tutorías y ejercitación. La mayoría de los estudiantes consideran que se necesita que los docentes tengan muy buena pedagogía, excelentes conocimientos y suficiente experiencia para mejorar el proceso enseñanza-aprendizaje del curso.

Buscando posibles soluciones a este problema se han llevado a cabo algunas estrategias de acompañamiento por parte del departamento de sistemas, ellas son: La implantación de asesorías académicas (denominadas monitorías estudiantiles) ofrecidas por estudiantes de cuarto semestre o superior a los estudiantes del curso de POO (en segundo semestre) y una constante motivación a los estudiantes para que hagan uso de las horas de consulta y el desarrollo de talleres y ejercicios fuera de clase.

La puesta en práctica de las anteriores actividades y el uso de diferentes estrategias metodológicas, por parte de los docentes que orientan la asignatura de POO, ha resuelto en parte algunos de los problemas expuestos. Una de las estrategias que mejores resultados ha presentado consiste en la ejercitación permanente y las actividades grupales que tienen como objetivo resolver problemas de POO. Se ha observado que para obtener mejores resultados a nivel general en la asignatura, es clave hacer énfasis en el diseño de diagramas de clase, debido a que es un tema fundamental para entender luego el manejo dinámico de objetos, diagramas de colaboración, mensajes y polimorfismo.

Teniendo en cuenta que las estrategias grupales han tenido mejores resultados, se decidió iniciar un proceso de investigación en Aprendizaje Cooperativo (AC) [2][3]aplicado a la edición de diagramas de clase, y que este trabajo pueda ser desarrollado en forma manual (tradicional) o en un ambiente de Aprendizaje Cooperativo Soportado por Computador (Computer Support for Cooperative Learning, CSCL) [4][5].

Al comenzar la investigación, aunque no se encontró un proyecto de iguales características, sí se encontraron iniciativas a nivel nacional e internacional que dieron aportes significativos para el inicio del trabajo, ellas son:

- Algebra Jam [6]: Es un ambiente de aprendizaje cooperativo que permite a grupos de estudiantes sincrónicamente resolver problemas que involucren modelos algebraicos y de forma remota. El sistema proporciona un juego de herramientas para ayudar a los estudiantes a establecer un objetivo en común y mantener el enfoque del grupo. Los aportes más significativos se relacionan con el manejo de modos de operación individual y en equipo incluyendo la notificación de las diferencias entre los integrantes del grupo, una pizarra de equipo, salón de charla, y el uso de roles (Observador, Aprendiz, Especialista, Líder, Preparador) para la resolución de problemas.

- Computational Analysis of Knowledge Sharing [7][8][9]: Es un ambiente CSCL que permite a grupos de estudiantes solucionar cooperativamente problemas que involucre el concepto de orientación a objetos utilizando la

técnica OMT (Object Modeling Technique propuesta por Jim Rumbaugh). Este sistema cuenta con un espacio de trabajo compartido y una interfaz de comunicación donde los estudiantes pueden construir en común un diagrama, mientras que están trabajando cooperativamente, existe un motor de análisis funcionando dinámicamente, determinando la interacción y recomendando acciones a un agente que idealmente se encuentra en línea supervisando a los estudiantes mientras que están aprendiendo. Este trabajo es importante porque muestra una forma estructurada de comunicación, que es el eje principal de la investigación.

- Ambiente Computacional de Ejercitación Colaborativa para Programación Estructurada [10][11]: Es una investigación que construye un método de ejercitación colaborativa diseñado según los métodos de aprendizaje colaborativo especializados en la solución de problemas y un ambiente computacional que soporta este método. Como conclusión de dos experimentos se observó un mejor rendimiento académico de los estudiantes de Informática (Computer Science), además de una excelente motivación e interés en usar este tipo de aplicaciones por parte de los estudiantes. Esta investigación es muy importante por el contexto teórico recopilado, los conceptos usados para la formulación del modelo, las herramientas usadas para la implementación del ambiente CSCL y la planeación, ejecución y análisis de los experimentos.

- The Collaborative Multi-User Editor Project IRIS [12]:En esta investigación se identifica y se satisface las necesidades reales para la edición colaborativa en red; Además se propone un ambiente flexible e integrable para la edición en grupo llamado IRIS, el cual presenta ventajas como: puede usarse en redes de área ancha, soporta los puestos de trabajo desconectados así como los puestos de trabajo móviles con bajo ancho de banda para establecer enlaces de comunicación (El usuario o la aplicación de interfaz de usuario puede determinar cuándo distribuir las actualizaciones y cuándo recibir las actualizaciones), soporta la colaboración síncrona y asíncrona, soporta la edición colaborativa de gráficos y finalmente, permite la importación y exportación de herramientas.

A continuación en la sección 2, se describe una técnica o método de aprendizaje cooperativo denominado "Cooperar" que tiene como objetivo orientar el desarrollo de ejercicios cooperativos de edición de diagramas de clase, en forma manual, es decir en el aula de clase o fuera de ella, sin soporte por computador. En la sección 3, se muestra el diseño general de un ambiente computacional para la edición cooperativa de diagramas de clase con UML denominada "SCU", y las características que lo hacen original y útil. Finalmente, se presentarán algunas conclusiones y el trabajo futuro que desarrollará el grupo de investigación con respecto a este tema.

2. MÉTODO DE APRENDIZAJE COOPERAR

Para facilitar la comprensión del método/técnica/estructura desarrollado por los autores de este artículo, primero se realiza una descripción introductoria, posteriormente se exponen las fases que desarrollan los docentes y los estudiantes en el método y finalmente se muestra una síntesis de los roles que pueden tomar los estudiantes.

2.1 Descripción

COOPERAR es un método de aprendizaje cooperativo que no es libre de contenido, empleado de forma particular para realizar actividades de edición cooperativa, donde los estudiantes realizan diagramas de clase con UML utilizando los conceptos de programación y modelado orientado a objetos que subyacen a dichos diagramas. Este método es cooperativo, más que colaborativo, debido a que estructura/define un conjunto de procesos/procedimientos que ayudan a la gente a interactuar en forma conjunta par cumplir con un objetivo general, que en este caso, consiste en aprender un contenido especifico, resolver un problema, completar una tarea o desarrollar una habilidad de modelado. En contraste, el aprendizaje colaborativo según Panitz [3] corresponde a una filosofía de interacción donde los aprendices son responsables de sus acciones, incluyendo el aprendizaje, y respetan las habilidades y contribuciones de sus compañeros sin ninguna técnica mediadora planteada por el docente. En la Tabla 1 se muestran las características más importantes del aprendizaje cooperativo y del cooperativo según Panitz.

Con COOPERAR se persiguen objetivos académicos y otros relacionados con el desarrollo de habilidades cooperativas. Los objetivos académicos consisten en que entre todos los estudiantes de un grupo planteen un modelo que dé solución a un problema o a un requerimiento de un sistema, entiendan la estrategia que se utiliza para editar y diseñar las clases y apliquen los conceptos vistos en clases previas. Los objetivos relacionados con las habilidades cooperativas ofrecen oportunidades a los estudiantes para cultivar habilidades sociales y comunicativas necesarias para el trabajo en equipo. Algunas habilidades sociales pueden ser: llegar a conocerse y confiar en el otro, comunicarse con precisión y sin ambigüedades, aceptarse y apoyarse, y resolver sus conflictos de manera constructiva [13].

En este método los estudiantes se dividen en grupos de dos a cuatro miembros. El docente entrega a todos los estudiantes uno o más ejercicios para que los estudien y se den una idea inicial del trabajo. Después los estudiantes se reúnen en sus grupos y en forma cooperativa dan aportes, escuchan los aportes de sus compañeros, discuten y solucionan

los ejercicios. La interacción entre los miembros del grupo puede realizarse basado en roles y sin estos (secuencial sin roles); cuando la interacción es basada en roles (Evaluador, Observador, Líder) se organiza la participación de todos los integrantes asignándole un rol a cada integrante del grupo, cuando la interacción es secuencial sin roles los integrantes del grupo puedan realizar cualquier tarea de forma síncrona o asíncrona. Después, el docente da un tiempo (dependiendo de la tarea en horas o días) para que realicen la tarea, hagan una evaluación de su trabajo cooperativo y finalmente todos los estudiantes presentan una evaluación individual del tema. En la Figura 1 se muestran las fases principales del método y en la Tabla 2 se puede apreciar un resumen de las características del método, de acuerdo a la propuesta de Slavin [14].

Aprendizaje Cooperativo	**Aprendizaje Colaborativo**
Es una estructura (técnica o método) predefinida de interacción y libre de contenido.	Es una filosofía de interacción, un estilo de vida personal, libre de estructuras y contenido.
Es un enfoque muy centrado o controlado por el docente.	Es un enfoque muy centrado en el estudiante, donde docente y estudiante comparten la autoridad y el control del aprendizaje.
Los individuos son orientados por una estructura que permite el logro de una meta o producto final específico, generalmente un contenido.	Los individuos son responsables de sus acciones, incluyendo su aprendizaje, y respetan las habilidades y contribuciones de sus compañeros.
Requiere un bajo grado de sofisticación de los estudiantes para trabajar en grupo. Es recomendado para enseñar a los estudiantes los procesos básicos de interacción social y el conocimiento de temas de carácter fundamental.	Requiere un alto grado de sofisticación de los estudiantes para trabajar en grupo. Es recomendado para extender las habilidades de pensamiento crítico y razonamiento, así como el mejor entendimiento de las relaciones sociales.
Hace énfasis en los procesos motivacionales, definidos por el método o la actividad.	Hace énfasis en los procesos cognitivos y meta cognitivos. La motivación es intrínseca.
Hace énfasis en los pasos y/o tareas específicas que se deben desarrollar en la actividad.	Hace énfasis en un proceso general que impacte la personalidad del individuo.

Tabla 1. Características principales del aprendizaje cooperativo y colaborativo según Panitz

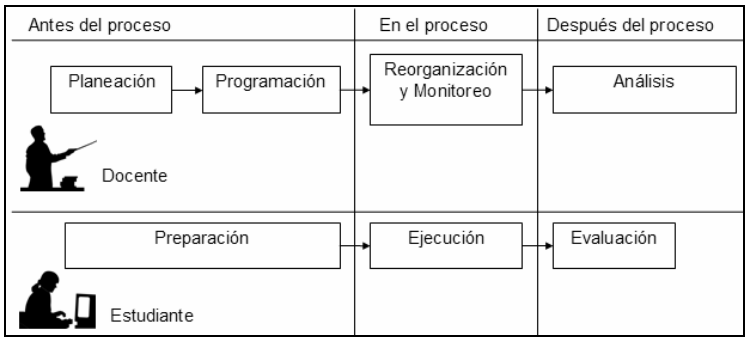


Figura 1. Fases principales de COOPERAR.

Nombre del Método	COOPERAR
Objetivos Grupales	SI
Responsabilidad Individual	SI
Iguales posibilidades de éxito	SI
Competencia de equipos	NO (el docente puede estimular este tipo de actividades)
Especialización en la tarea	NO
Adaptación a las personas	NO

Tabla 2. Características de COOPERAR.

2.2 Fases que desarrolla el Docente

El docente tiene cuatro fases principales. La primera fase es la de **planeación** la cual es previa a la actividad y en la que formula los casos a modelar (ejercicios), los tiempos, entre otros. La segunda fase es la de **programación**, en la que organiza de forma tentativa los grupos de ejercitación, al azar o por otro criterio. La tercera fase es la de **reorganización** y el **monitoreo** de los grupos, iniciando con la reorganización de los grupos si esto se requiere, entregando los materiales y dando tiempo a los estudiantes para que lean los casos a modelar, luego monitorea los grupos observando los aportes de los estudiantes, el estado de solución del problema, verificando que se estén cumpliendo las habilidades

cooperativas y reorienta a los grupos que se encuentren desenfocados, entre otras cosas. Y la última fase es la de un **análisis** general de la actividad con el fin de mejorar el desarrollo de futuras actividades y dar recomendaciones a los estudiantes sobre situaciones observadas que fueron acertadas o que deben ser mejoradas. El docente debe realizar estas fases para cada actividad que vaya a desarrollar en un curso, o reutilizar algunos productos en dos o más cursos, como por ejemplo los productos de la fase de planeación. En la Figura 2 se puede observar una síntesis de las actividades que desarrolla el docente en cada fase.

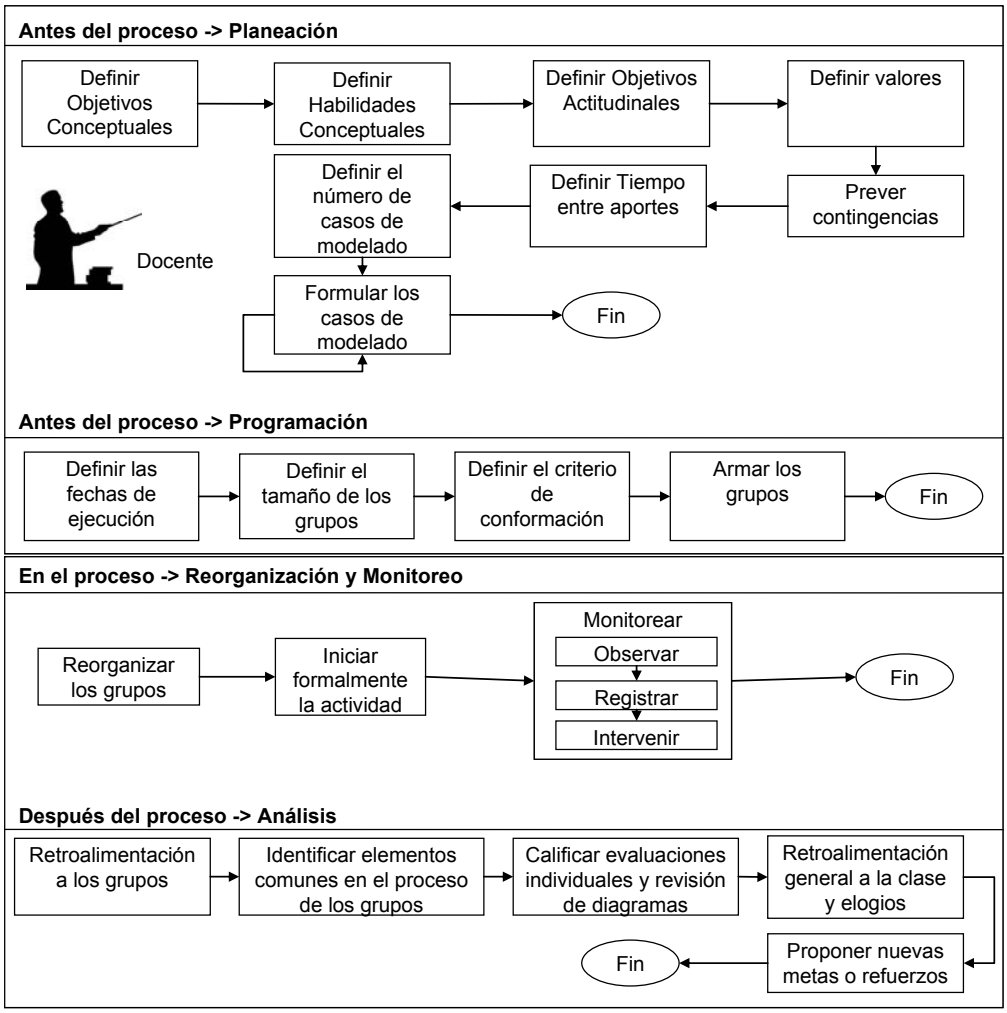


Figura 2. Actividades desarrolladas por el Docente.

2.3 Fases que desarrolla el Estudiante

Se han definido tres fases para el estudiante. La primera fase es la de **preparación** previa en la que el estudiante lee, repasa, analiza, sintetiza y se ejercita individualmente sobre los temas vistos en clase. La segunda fase es la de **ejecución** de la actividad en sí misma, donde realiza la solución de los ejercicios (construcción de los diagrama de clases) con sus compañeros de grupo. Y la tercera fase es donde realiza la **evaluación** del procesamiento del grupo y de sus propios conocimientos con un examen individual. En la Figura 3 se muestra una síntesis de las actividades que desarrolla el estudiante en cada fase.

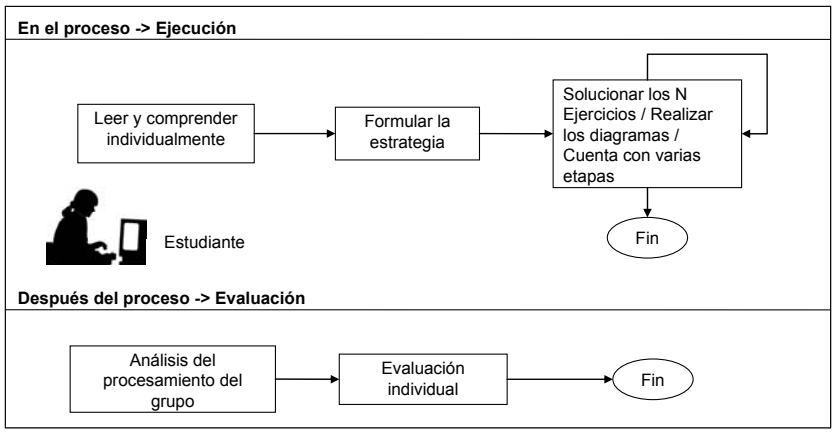


Figura 3. Actividades desarrolladas por el Estudiante.

Cuando la interacción en la fase de ejecución se realiza basada en roles, cada integrante del grupo cumple una función específica, debe realizar aportes y cumplir con los objetivos propios del rol. En la ejecución de la actividad, se realizan cambios de roles cada vez que ocurra un cambio de etapa (por ejemplo: pasar de la etapa de realizar clases a la etapa de realizar atributos dentro de una clase, se habla de etapas sólo en la fase de ejecución). Estos cambios buscan que cada estudiante desempeñe los diferentes roles en el desarrollo de cada ejercicio, de manera proporcionada y equitativa. El tamaño de los grupos es de dos (2) a cuatro (4) personas, donde el rol por defecto es el de **Diseñador**, encargado de dar aportes en la solución del problema, o cumplir temporalmente (en cada etapa) con uno de los siguientes roles:

- **Líder**: Se ocupa de iniciar y terminar las sesiones de lluvia de ideas en la cual los integrantes del grupo (incluido él) generan aportes para identificar los elementos que hacen parte de la solución del problema como son: las clases, los atributos, los métodos, las asociaciones, la navegabilidad de las asociaciones y las relaciones de dependencia entre las clases. Realiza un consenso en el grupo pidiéndoles a los integrantes que justifiquen el por qué de sus aportes y los aportes realizados por sus compañeros en la fase de lluvia de ideas. Establece un espacio para la votación de las ideas seleccionando los elementos que serán utilizados y posteriormente graficarlos. Por último realiza el cambio de etapa, con lo cual también se logra el cambio de rol (por ejemplo: pasar de clases a atributos, a asociaciones etc.). Su fin principal consiste en coordinar las actividades del grupo.

- **Evaluador**: Esta persona tiene como objetivo verificar el cumplimiento de las metas académicas fijadas en la tarea, proporciona un soporte para el docente en el proceso evaluativo del grupo, realiza preguntas para verificar que los integrantes del grupo están comprendiendo la actividad en cada etapa (clases, atributos, métodos, asociaciones, navegabilidad de las asociaciones y las relaciones de dependencia entre las clases), luego realiza una evaluación teniendo en cuenta lo aportes de cada integrante del grupo, incluidos los propios y por ultimo aconseja al líder en el desarrollo de la actividad. Su fin principal consiste en buscar que todos los miembros del grupo entiendan el problema y su solución.

- **Observador**: (Opcional) Esta persona tiene como objetivo verificar el cumplimiento de las habilidades cooperativas fijadas en la tarea (si se planean), proporcionando un soporte para el docente en el proceso evaluativo de los grupos. Da aportes, analiza el diálogo durante la interacción de los integrantes para verificar el cumplimiento de las habilidades cooperativas propuestas por el docente. Verifica que la clasificación de los mensajes sea correcta (por ejemplo: sugerencia, crítica, etc.). Da sugerencias para mejorar el uso de habilidades cooperativas. Realiza una calificación teniendo en cuenta aspectos cooperativos y por ultimo aconseja al líder en el desarrollo de la actividad.

En la fase de ejecución, los cambios de tareas están dados por los siguientes momentos: definición de clases a nivel general, definición de atributos, definición de métodos, definición de asociaciones, definición de navegabilidad, definición de multiplicidad y definición de las relaciones de dependencia. De esta forma se logra que la fase sea muy organizada y siguiendo una metodología de diseño recomendada por diferentes autores (por ejemplo Larman [15], Booch, Rumbaugh Jacobson [16][17]). Esos momentos se pueden repetir varias veces en un proceso de refinamiento que busque en cada ciclo una mejora del producto del ciclo previo.

Cuando la interacción se planea para que se realice sin roles, es decir secuencialmente, se precisa que el docente defina previamente un tiempo mínimo de espera para realizar aportes en los diagramas de clase, con ello, se busca dar un

tiempo para que los estudiantes expliquen a sus compañeros, formulen preguntas, entre otros, y se evita que sólo un estudiante se dedique a resolver el problema, sin ayudar y/o preguntar a sus compañeros.

3. AMBIENTE CSCL PARA "COOPERAR" - SCU

SCU (**S**olución informática para la edición **C**ooperativa de diagramas de clase con **U**ml) es un ambiente computacional (herramienta software) de edición cooperativa de diagramas de clase fundamentado en el método COOPERAR descrito anteriormente. SCU se encuentra en fase de construcción y dentro de sus requisitos principales se encuentran los siguientes:

- Integrar las fases de método en un ambiente único. Cada tipo de usuario (estudiante y docente) tendrán su propia interfaz de usuario y en ella podrán identificar claramente las fases y las actividades del método.
- La ejecución de una actividad debe poder realizarse en forma sincrónica y asincrónica.
- El editor de diagramas de clase debe cumplir con la versión 1.5 de UML.
- El editor de diagramas debe realizar una actualización de los aportes de los compañeros en máximo un (1) segundo. Con lo anterior se busca una mejor comprensión y visualización del trabajo que hacen los compañeros de grupo.
- Se debe contar con un sistema de comunicación (Chat/foro) de los estudiantes del grupo en forma privada o pública, y de los estudiantes con el docente. El docente puede comunicarse con cualquier grupo, con un estudiante en particular o con toda la clase. Esta comunicación les permitirá compartir su conocimiento y discutir sus ideas de diseño y trabajo en equipo de una manera estructurada, que facilite la comprensión de los mensajes entre los miembros del equipo. Todos los mensajes deben quedar registrados en la bitácora del ambiente.
- La comunicación se clasificará en Mensajes de Aceptación, Negación, Pregunta, Sugerencia y Varios, buscando con lo anterior disminuir la ambigüedad del idioma español, en donde frases como "Qué hiciste" se podría interpretar como un corrección de alguna acción mal hecha (como se te ocurre hacer eso), cuando lo que se quería era una explicación de la acción o viceversa.
- El diseño de la interfaz debe seguir los parámetros establecidos por expertos del diseño como la importancia de la creación de interfaces simples, entendiendo lo que podemos y no podemos hacer, los mapas del pensamiento humano que presenta la ciencia de la psicología cognoscitiva y conocer los aportes de la ingeniería cognética. La comprensión que sólo tenemos un centro de atención y al reconocimiento del papel central que juega la naturaleza de la formación de hábitos en como reaccionamos ante varias metodologías de interfaz [18]. Estos parámetros nos dejan claro que debemos evitar al máximo los modos, que aunque son reconocidos desde hace mucho tiempo como indeseables, se encuentran presentes en la gran mayoría de las interfaces de hoy en día, y cualquier solución que se quiera dar para obtener una interfaz ideal debe carecer de modos, ser lo más monótona posible (que tiene una forma de efectuar una tarea) y no hacer distinciones entre usuarios expertos y novatos promoviendo la uniformidad entre usuarios. Además se aplicarán los principios de diseño de interfaces como son: Visibilidad, Uso, "Affordance", Modelo Conceptual, Modelo Mental, Coincidencia "Mapping" y Feedback.

3.1 Servicios agregados a SCU

Uno de los motivos por el cual los docentes se muestran renuentes a aplicar el aprendizaje cooperativo en las aulas de clases es la sobrecarga de trabajo ya que "el docente decide los objetivos, toma una cantidad de decisiones previas a la enseñanza respecto del tamaño del grupo y los materiales necesarios, explica la tarea y la estructura de los objetivos cooperativos a sus alumnos, los controla mientras trabajan, interviene cuando es necesario, evalúa el aprendizaje de sus alumnos y se asegura de que los grupos procesen el grado de eficiencia con el que están trabajando"[13].Además si le agregamos el computador y los problemas que implica para el docente, como el hecho de que sus estudiantes pueden estar ubicados en distintos puntos geográficos y el que no va a poder monitorear a todos los grupos al tiempo como lo podría hacer en un salón de clase, le dan motivos suficientes al docente para no querer cambiar de método y seguir con sus clases tradicionales.

Con la revisión de algunos trabajos relacionados, se encontraron dos alternativas que alivian el proceso de monitoreo y evaluación de los estudiantes, que es realmente la carga más grande para el docente en la aplicación del aprendizaje cooperativo soportado por computador, ellas son:

- El registro histórico de las conversaciones: Aunque proporciona una idea de la interacción entre los participantes de la actividad cooperativa, al docente le es casi imposible recrearla, además el análisis "manual" de esta información es muy compleja si la actividad que se realizó tuvo una duración considerable o la participación de muchos estudiantes en varios grupos.

- Agentes o Tutores Inteligentes: Existen varias modalidades, el agente o tutor inteligente se encarga de ofrecer ayuda y retroalimentación en ciertas acciones y ayuda con ello a resolver problemas, en otros casos registra los problemas individuales y de grupo y en otros permiten la creación de bancos para desarrollar teorías de grupo. Si

bien estos agentes son de notable ayuda, muchas veces estos no son precisos ya que muchos de estos no abarcan las innumerables apreciaciones que un docente realizará cuando analiza los datos, de forma cuantitativa, cualitativa y subjetiva.

Teniendo en cuenta que estas opciones tienen sus limitaciones, se vio la necesidad de crear nuevos servicios, ellos son:

- Estadísticas de aportes: Este servicio permite realizar un registro dinámico de las participaciones de cada integrante de un grupo, distinguiendo cada uno de estos por un color diferente, con el fin de que a simple vista el estudiante o profesor puedan mirar en la pizarra el número de aportes que ha realizado cada miembro del grupo, además se presentará una gráfica en barras donde se podrá establecer porcentualmente el aporte de cada estudiante clasificándolos en: número de clases, atributos, métodos, asociaciones, navegabilidad de las asociaciones, y relaciones de dependencia entre las clases, que el estudiante crea, elimina, modifica o que un compañero le ha cambiado (por ejemplo un contador de "mis clases que han sido borradas/modificadas"). Este servicio le proporciona una retroalimentación constante al estudiante y se espera que con ella se estimule a los estudiantes rezagados, buscando que el grupo los ayude y que el estudiante trate de estar al nivel de grupo, contribuyendo de esta forma a mantener la interdependencia positiva, la promoción de la interacción, la responsabilidad individual, el desarrollo de habilidades interpersonales y de grupos pequeños, y el procesamiento individual (pilares del aprendizaje cooperativo), y al tener esta retroalimentación dinámica se le permite tanto al estudiante como al grupo mejorar y corregir errores en el desarrollo de la actividad y no dejarlos como conclusión al final de la actividad agilizando el proceso de maduración del grupo cooperativo.

- Reproducción de la Actividad: Una de las dificultadas que enfrenta el docente en el momento de evaluar una actividad cooperativa, es la de evaluar el proceso y no sólo el producto final que le presenta el grupo cooperativo[19]. En ciertas ocasiones se precisa ver el aporte que hizo cada integrante en el desarrollo de la actividad y aunque las estadísticas dinámicas le brindan información detallada de la forma de trabajar del grupo cooperativo, sigue existiendo la necesidad de que el docente puede recrear los hechos en el orden en que sucedieron de una manera precisa. Con el objetivo de dar una solución a este problema, se diseñó un servicio capaz de almacenar todas las acciones que realiza el grupo cooperativo con el fin de que el docente pueda recrear o reproducir como un "video" lo ocurrido en el desarrollo de la actividad cooperativa. Es importante resaltar que el docente tiene la opción de realizar una reproducción de la actividad (video) a diferentes velocidades según su conveniencia (normal, 2x, 3x, etc.). Para este servicio, se descartó desde el principio el uso de herramientas de grabación de videos como Snagit (http://www.techsmith.com/), debido al gran volumen de almacenamiento necesario y a la disminución en el rendimiento de las aplicaciones. Se utiliza una idea similar al Log de Transacciones de un motor de bases de datos.

3.2 Diseño general de SCU

Con base en los requerimientos planteados para SCU, y teniendo en mente que las estadísticas, el registro y la posterior reproducción de la actividad son un factor clave para el éxito del ambiente computacional SCU, se inició con una serie de ciclos cortos de diseño, construcción y prueba de prototipos que permitieron llegar a la arquitectura que se presenta en la Figura 4.

En la arquitectura se pueden observar tres capas (separadas por líneas punteadas), ellas son el almacenamiento (Oracle), la lógica de la aplicación compuesta por un conjunto de Servicios Web XML en Internet Information Server (IIS) y un conjunto de clases que prestan servicios remotos de Comunicación y Coordinación (Servidor de Coordinación y Registro de Transacciones y de Conversación) y finalmente la interfaz de usuario diseñada para aplicaciones Windows, en la que se presentan los formularios con los que interactúan los estudiantes y los profesores. Cada usuario tiene su propia aplicación de interfaz, es decir, existe una aplicación que usan los profesores y otra aplicación que usan los estudiantes. En cada una de las capas se utilizarán distintos patrones de software, como por ejemplo, el modelo-vista-controlador, el de indirección, el singleton, el table gateway, entre otros.

La comunicación entre las aplicaciones cliente y los servicios Web XML se realizan a través de SOAP, y las tareas principales de estos servicios se relacionan con el manejo de la autorización, la autenticación, el despliegue de los contenidos digitales desarrollados para el curso, la revisión de datos históricos, entre otros. El trabajo más interesante lo ofrecen los objetos distribuidos con Remoting que permiten la comunicación y coordinación de las actividades. A través de estos objetos se informan los cambios en un diagrama de clases a los compañeros que trabajan síncronamente en una actividad determinada. Para dar mayor velocidad a la comunicación estos objetos se comunican con TCP y los datos que se envían son binarios. El objeto Servidor de Coordinación se encarga de registrar en la base de datos los cambios en la comunicación (Chat) y en el diagrama de clases e informar a los clientes que trabajan en el mismo diagrama; para ello usa mensajes asincrónicos y múltiples hilos de ejecución. El objeto Cliente de Coordinación se encarga de informar los

cambios en el Chat (Registro de la Conversación) y en el Diagrama de Clases (Registro de Transacciones en el diagrama) al Servidor y de realizar los cambios en los objetos de lógica e interfaz de la aplicación Cliente.

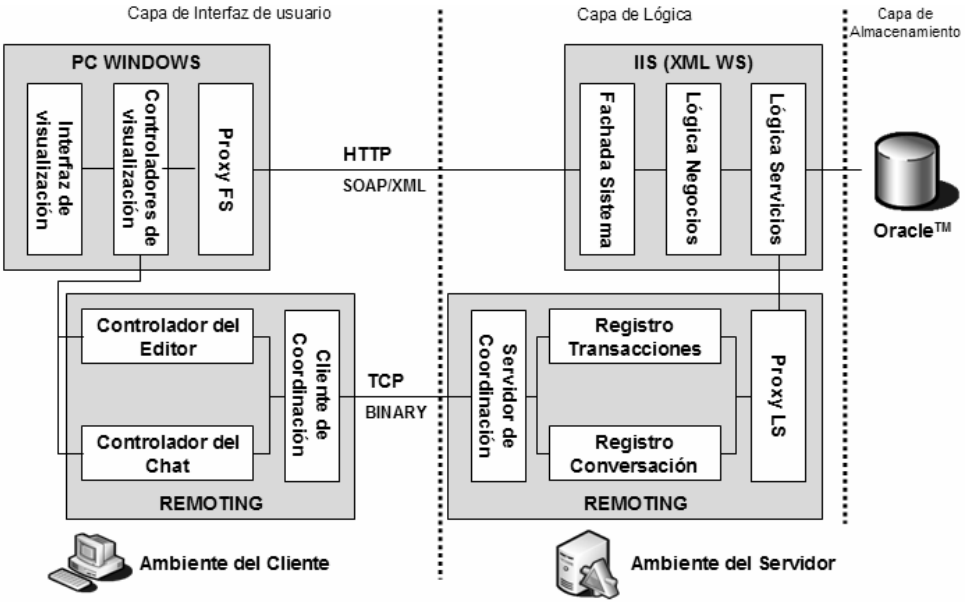


Figura 4. Arquitectura general de SCU.

En la Figura 5 se listan los pasos generales que se realizan para el registro de los cambios en los diagramas de clases y el informe inmediato a los compañeros de grupo que están trabajando sincrónicamente. Estos pasos son:

- 1) Un estudiante realiza un cambio en el diagrama (por ejemplo, crea una nueva clase),
- 2) El Cliente de Coordinación a través del Controlador del Editor traduce ese cambio en una "instrucción" corta pero detallada (por ejemplo, nombre, posición, ancho, alto) y la informa al Servidor de Coordinación,
- 3-6) Se encargan de registrar los cambios en las tablas correspondientes y si no se presenta ninguna inconsistencia se le informa al Servidor de Coordinación (a través del registro de Transacciones) que la operación ha sido exitosa,
- 7) El Servidor de Coordinación informa a los Clientes de Coordinación de los compañeros de grupo que se realizó una "instrucción" específica y al Cliente que realizó el cambio que todo se ejecutó con éxito,
- 8) El Controlador del Editor traduce la "instrucción" y modifica la interfaz del editor con el cambio (por ejemplo, dibuja la nueva clase en la posición y con el tamaño informado).

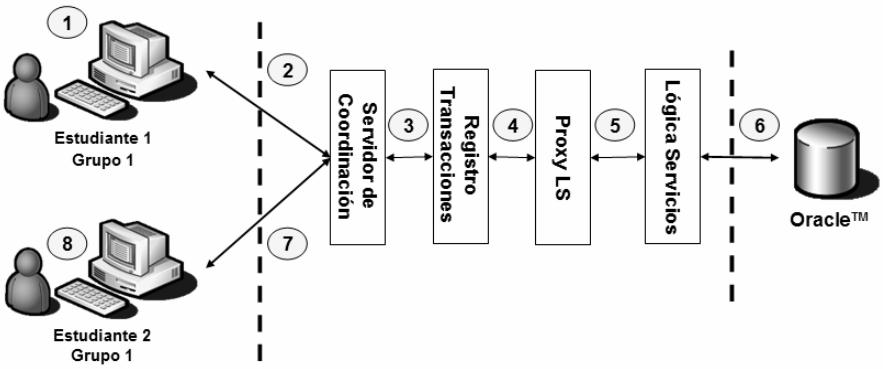


Figura 5. Pasos en el registro y difusión de los cambios en el diagrama de clases.

En la Figura 6 se presenta un diagrama general del despliegue de SCU en un ambiente donde sólo se cuenta con dos servidores, en este caso uno de base de datos y uno de aplicación. Es preciso señalar que los componentes de la lógica de la aplicación se pueden dividir físicamente en más servidores.

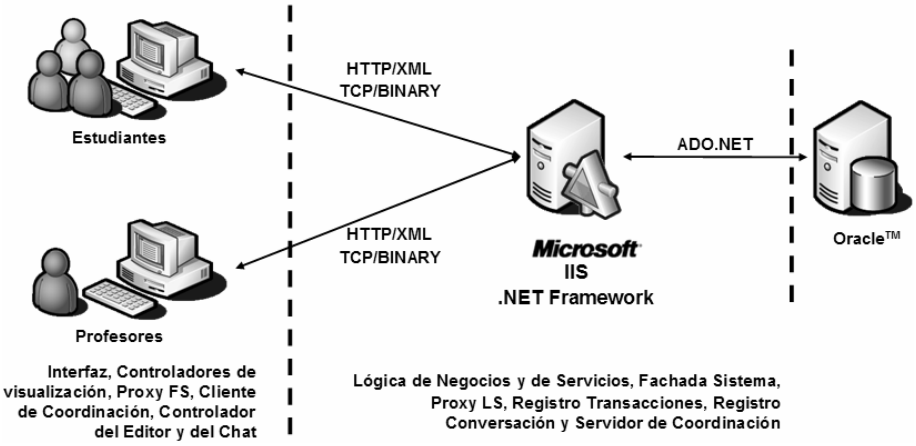


Figura 6. Despliegue de SCU en dos servidores y n clientes.

La reproducción de la actividad se basa en el Log de Transacciones del diagrama, un temporizador y el Controlador del Editor (mencionado en el paso 8 de la Figura 5) que se encuentra en la interfaz de usuario del profesor. Con estos componentes, la tarea se reduce en general a los siguientes pasos:

- 1) Iniciar el temporizador y usar la velocidad seleccionada por el profesor, es decir tiempo real (normal), o más rápido (2x, 3x, etc.),
- 2) Leer el Log de Transacciones en orden de fecha y hora. Comparar la fecha de la operación más reciente con el tiempo transcurrido en el temporizador, si no hay correspondencia, esperar un momento más,
- 3) Si hay correspondencia entre la fecha y hora de la instrucción con el tiempo transcurrido en el temporizador, solicitar al Controlador del Editor que traduzca la "instrucción" y modifique la interfaz del editor con el cambio en el dibujo (por ejemplo, dibujar la nueva clase en la posición y con el tamaño informado).
- 4) repetir las operaciones 1 a 3 hasta que se acaben las operaciones/instrucciones en el Log, o el profesor detenga el video.

Actualmente el ambiente computacional se encuentra en desarrollo y se planea desarrollar un par de pre-experimentos con grupos de Programación Orientada a Objetos a principios de agosto de 2005, con el objetivo de obtener algunos indicios que le permitan al grupo de investigación profundizar el trabajo o reorientarlo.

4. CONCLUSIONES Y TRABAJO FUTURO

COOPERAR es un método que está diseñado para realizar ejercitación presencial en el aula y fuera de ella, pero gracias al ambiente computacional SCU se extienden las capacidades de COOPERAR, llevándolo a escenarios de la educación a distancia soportada por computador, actualmente denominada Educación en Línea.

Auque en el diseño de Cooperar se estimula la presencia del conflicto socio-cognitivo[20], ya que el trabajo gira alrededor de la solución de problemas en un área donde el problema tiene muchas formas de ser resuelto acertadamente y bajo un cierto orden (etapas en roles o tiempos de espera sin roles), se hace preciso determinar en la práctica si el nivel de ese conflicto es manejable en el aula y en el computador, tema que debe ser cuidadosamente estudiado en los experimentos.

La creación de los servicios de estadísticas dinámicas, registro y reproducción de actividades aunque ya han sido propuestos por algunas investigaciones, en este proyecto se plantea una alternativa distinta que busca disminuir la carga del docente, facilitándole el proceso de monitoreo y evaluación de las actividades, sin impactar negativamente en el rendimiento de la aplicación, ni almacenando grandes volúmenes de información.

En el diseño arquitectónico del ambiente computacional SCU se aplicaron varios conceptos de ingeniería de software y patrones, pero uno de los logros más importante hace referencia a la posibilidad de extender la implementación a diversos tipos de clientes (no sólo computadores personales), como Tablet PCs, asistentes digitales personales (como Pocket PC), entre otros. Con ello se da la oportunidad de probar el método Cooperar y el ambiente computacional SCU en un ambiente de Aprendizaje Cooperativo Soportado por Computador Móvil (Mobile CSCL). Trabajo que también espera desarrollar el Grupo de Investigación en Tecnologías de la Información (GTI) de la Universidad del Cauca.

Agradecimientos

Este trabajo fue cofinanciado por COLCIENCIAS a través de su proyecto de I+D en ETI titulado "Unicauca Virtual Fase II" con código 1103-14-14897. También agradecemos a todo el grupo de profesores y estudiantes que participan en el proyecto por sus oportunos aportes y comentarios.

References

[1] Muñoz, C. And Salazar, J. Informe del Encuesta a estudiantes y profesores de Programación Orientada a Objetos Universidad del Cauca, Departamento de Sistemas, 2004.

[2] Dillenbourg, P.et al. The evolution of Research on Collaborative Learning In H. Spada and P. Reimann (Eds) Learning in Humans and Machines. Elsevier. 1996.

[3] Panitz, T. Collaborative versus Cooperative Learning- A comparison of the two concepts which will help us understand the underlying nature of interactive learning. Cooperative Learning and College Teaching, V8, No. 2, Winter 1997. ver Ted's Cooperative Learning e-book en http://home.capecod.net/~tpanitz/ebook/contents.html.

[4] Roschelle, J. & Teasley, S. The construction of shared knowledge in collaborative problem solving. In C.E. O'Malley (Ed) Computer- supported collaborative learning. Heidelberg: Springer-Verlag (in press).

[5] Barbara, W. Computer Supported Collaborative Learning, An Overview. Lecture notes from IVP 482, University of Bergen, 1998.

[6] Singley, M. et al. Algebra Jam. Proceedings of the ACM conference on Computer supported cooperative work. Philadelphia, Pennsylvania, United States. Pages: 145 – 154. ISBN:1-58113-222-0, 2000.

[7] Soller, A.. CSCL 2002 Workshop Designing Computational Models of Collaborative Learning Interaction. University of Pittsburgh, 2002.

[8] Soller, A. et al. Promoting Effective Peer Interaction in an Intelligent Collaborative Learning System. In Proceedings of 4th International Conference on Intelligent Tutoring System. San Antonio, TX., 1998, pp. 186-195.

[9] Soller, A. et al. What makes peer interaction effective? Modeling effective communication in an intelligent CSCL. American Association for Artificial Intelligence. www.aai.org, 1999.

[10] Cobos, C. Ambiente computacional de ejercitación colaborativa para la solución de problemas utilizando los conceptos de programación estructurada. Tesis de maestría en Informática, Gomez, L. (Director), UIS, 2003.

[11] Cobos, C. et al. Ejercitación Colaborativa y Programación Estructurada. *Jornadas Chilenas de la Computación*. Arica, Chile, Noviembre 8 al 12 de 2004. ISBN: 956-7021-18-X.

[12] Koch, M. The Collaborative Multi-User Editor Project Iris TUM-I9524, Inst. für Informatik. Technische Univ. München, http://www11.informatik.tu-muenchen.de/publications/pdf/Koch1995.pdf, 1995.

[13] Johson, D. and Johson R. Aprender Juntos y Solos: Aprendizaje cooperativo, competitivo e individualista. Ed. Aique, 1999.

[14] Slavin, R. Aprendizaje Cooperativo: Teoría, investigación y práctica. Ed. Aique, 1999.

[15] Larman, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition). Pearson Education. 2002. ISBN: 0-13-092569-1.

[16] Jacobson, I., Booch, G., Rumbaugh, J. The Unified Software Development Process. Addison-Wesley. Pearson Education. ISBN: 0-201-57196-2.

[17] Booch, G. Object-Oriented Analysis and Design with Applications (2nd Edition). Addison-Wesley. Pearson Education. ISBN: 0-8053-5340-2.

[18] Raskin, J. Diseño de sistemas interactivos. La importancia de nuestra relación con las computadoras. Pearson Education, México. http://www.pearsonedlatino.com, 2001.

[19] Collazos, C., Guerrero, L.,Pino, J., and Ochoa, S., A Method for Evaluating Computer-Supported Collaborative Learning Processes; International Journal of Computer Applications in Technology, Vol. 19, Nos. 3/4, pp.151-161, 2004.

[20] Johnson, R. and Johnson, D. Handbook of Research for Educational Communications and Technology. Edited by David H. Jonassen. 1997.

Mecanismos de Flexibilização de Sistemas Gerenciadores de Workflow para Antecipação de Tarefas

Igor Steinmacher, Edson A. O. Junior
Universidade Estadual de Maringá, Departamento de Informática,
Maringá, Paraná, Brasil
igor@din.uem.br, edson @din.uem.br

Fabrício R. Lazilha
Centro Universitário de Maringá (CESUMAR), Departamento de Infomática
Maringá, Paraná, Brasil
fabricio@cesumar.br

José Valdeni de Lima
Universidade Federal do Rio Grande do Sul, Instituto de Informática,
Porto Alegre, Rio Grande do Sul, Brasil
valdeni@inf.ufrgs.br

Abstract

Traditional Workflow Management Systems (WfMS) are too rigid, posing problems when cooperation and human behavior are important factors. Under these circumstances, there is a need for a relaxation of the end-begin dependency. Task anticipation can provide this relaxation. Task anticipation consists in relaxing the flow of control and data information among tasks, allowing the exchange of results among sequential tasks even before the conclusion of the former task. This anticipation promotes a better cooperation and augments the performance in terms of process execution time. The goal of this work was to study mechanisms of task anticipation in Workflow Management Systems. The study resulted in the identification and classification of flexibilization and anticipation management strategies according to the phase of the life cycle where the flexibility is applied (definition time/execution time). In this sense, two strategies are specified*: a priori* and *a posteriori.* Through a priori management, changes are made on process model during process definition time. A posteriori approach changes the way that the workflow process is executed, triggering some tasks before their actual start time, without the real necessity of changing the process definition.
Keywords: Workflow, Anticipation, Flexibilization, Cooperation.

Resumo

Sistemas Gerenciadores de Workflow (SGWf) tradicionais são rígidos e inflexíveis, apresentando alguns problemas quando da necessidade de sua utilização no controle de processos com características cooperativas, em que o comportamento humano deve ser levado em conta. Este tipo de processo deve apresentar uma certa flexibilidade de forma a possibilitar a troca de resultados entre tarefas seqüenciais. A fim de possibilitar este tipo de comportamento, surge a antecipação de tarefas. A antecipação implica no aumento da cooperação entre tarefas e em melhor desempenho com relação ao tempo de execução dos processos. Este trabalho apresenta um estudo das abordagens de flexibilização de SGWfs, visando encontrar maneiras de prover antecipação de tarefas. O estudo resultou na identificação e classificação das estratégias de gerenciamento de antecipação de tarefas de acordo com a fase do ciclo de vida em que a flexibilização é aplicada (tempo de definição/tempo de execução). Neste sentido duas estratégias são especificadas: a priori e a posteriori. A primeira delas é totalmente vinculada à definição dos processos, sendo prevista no modelo de processo e a segunda está relacionada à execução dos processos, deixando a decisão por antecipar ou não antecipar para a fase de execução dos processos.
Palavras chave: Workflow, Antecipação, Flexibilização, Cooperação

1. INTRODUÇÃO

Sistemas Gerenciadores de Workflow (SGWf) têm atraído muita atenção nos últimos anos, por serem baseados em um modelo simples que permite definir, executar e monitorar a execução de processos. Mas este esquema é um tanto quanto rígido e inflexível, apresentando alguns problemas quando da necessidade de sua utilização no controle de processos com características cooperativas, em que o comportamento humano deve ser levado em conta [15], [16].

Segundo Aalst e Hee [1] esta inflexibilidade e rigidez dos produtos atuais acabam por espantar muitos usuários potenciais. Hagen e Alonso [6] apresentam o mesmo ponto de vista, dizendo que inflexibilidade e rigidez acabam por tornar as tecnologias de workflow atuais restritas à apenas alguns domínios de aplicação. Várias outras pesquisas apontam estes pontos como uma das grandes razões da resistência à utilização de SGWfs em alguns domínios de aplicação [8], [12], [13], [16].

Visto isto, fica evidente a necessidade de fornecer meios para relaxar algumas das restrições impostas pelos SGWfs tradicionais. Uma destas restrições é a impossibilidade das tarefas produzirem, disponibilizarem ou utilizarem resultados intermediários, sendo as tarefas em vistas como caixas pretas. Relaxando esta restrição, as tarefas poderiam ser disparadas antes mesmo da conclusão de suas antecessoras, ocorrendo a chamada antecipação de tarefas [4]. Este é um dos objetivos deste trabalho, encontrar maneiras de flexibilizar o modelo de workflow tradicional, de forma a possibilitar que tarefas seqüenciais possam trocar resultados intermediários, iniciando sua execução de maneira antecipada.

Com base nisto, este trabalho tem como objetivo apresentar um estudo sobre os meios de flexibilizar-se os SGWfs de modo a prover a antecipação de tarefas. O estudo resultou na identificação e classificação das estratégias de flexibilização e de gerenciamento de antecipação de tarefas. Esta classificação foi feita de acordo com a fase do ciclo de vida em que a flexibilização é aplicada (tempo de definição de processo / tempo de execução do processo). Duas estratégias são especificadas: *a priori* e *a posteriori*. A primeira delas é totalmente vinculada à definição dos processos e deve ser prevista no modelo. Esta forma de antecipação pode ser utilizada em qualquer SGWf existente, porém, não oferece flexibilidade pré-determinada, continuando este estático e exigindo grande esforço de modelagem. A outra maneira de prover antecipação está relacionada à execução dos processos, deixando a decisão por antecipar ou não antecipar para a fase de execução dos processos. Esta abordagem é mais complexa e exige alterações na máquina de execução do SGWf, porém, oferece grande flexibilidade ao processo e não exige esforço extra durante a modelagem.

O restante do artigo está organizado da seguinte forma: a seção 2 traz uma revisão bibliográfica dos trabalhos relacionados à flexibilização de SGWfs e uma classificação segundo a fase em que se dá tal flexibilização. A seção 3 apresenta a antecipação de tarefas. A seção 4 traz as possíveis maneiras de aplicá-la no contexto de SGWfs. E, finalmente, na seção 5 são apresentados as conclusões e trabalhos futuros.

2. FLEXIBILIDADE EM SISTEMAS GERENCIADORES DE WORKFLOW

A busca por aumento na flexibilidade em sistemas de workflow é um tema muito explorado em pesquisa na atualidade. Tais pesquisas têm por objetivo, basicamente, tornar sistemas gerenciadores de workflow úteis a todas as áreas em que estes podem ser aplicados. A flexibilidade buscada vem de maneira a acabar com a estrutura extremamente rígida encontrada nos sistemas de workflow tradicionais.

Esta seção vem apresentar as pesquisas e abordagens relacionadas à flexibilidade encontradas na literatura. A partir da análise das propostas encontradas, foi possível criar uma classificação básica do tipo de flexibilidade que as abordagens podem oferecer: flexibilidade *a priori* e flexibilidade *a posteriori*.

Flexibilidade *a priori* é aquele tipo de flexibilidade oferecida durante a modelagem dos processos. Este tipo de flexibilidade está relacionada ao tempo de definição de processos, criando meios de tornar a execução de um processo menos restritiva através de novas maneiras de modelar processos.

Já nas abordagens onde é proposta a flexibilidade *a posteriori* são apresentadas maneiras de tornar os SGWfs menos rígidos através de mudanças em tempo de execução dos processos. Estas mudanças dizem respeito a alterações dinâmicas na definição dos processos e alterações no modo de execução dos processos.

2.1 Flexibilidade *a priori*

A primeira abordagem a ser apresentada é a proposta do operador Coo [3], que foi desenvolvido a fim de tornar os modelos de workflow capazes de representar as interações entre tarefas cooperativas. Tal operador não cobre todos os aspectos da cooperação, sendo, principalmente, dedicado ao gerenciamento de cooperação entre tarefas. Um exemplo da utilização deste operador é a resolução de exercícios por um estudante e a correção dos mesmos por um professor em um ambiente de ensino a distância. Os resultados dos exercícios podem ser enviados ao professor a qualquer momento, a fim de obter-se correções e sugestões por parte do professor. Esta interação pode ocorrer quantas vezes for necessário. O professor não é obrigado a corrigir todos os resultados intermediários que lhe são enviados,

sendo obrigado apenas a corrigir a versão final dos exercícios. A criação de um operador para modelagem de tarefas que devem cooperar entre si é uma idéia muito interessante. Através deste operador pode-se mapear a interação entre tarefas, permitindo a troca de resultados e a antecipação de tarefas. Porém, a necessidade de criação de novos elementos de modelagem não é muito interessante. Porém, a complexidade inserida à modelagem dos processos com as alterações propostas no Coo é grande, exigindo do responsável pela definição dos processos a configuração prévia dos meios com que se deve dar a troca de informações entre as tarefas.

Outra proposta de flexibilidade a priori é encontrada no Freeflow [2]. Esta abordagem também possibilita relacionamentos diferenciados entre tarefas. Porém os relacionamentos propostos no Freeflow são declarativos, capazes de separar as dependências entre tarefas (informações do sistema) e as informações do usuário. Isto é feito através de um modelo de estados para as tarefas que distingue entre estados do usuário e estados do sistema.
Nesta abordagem existe apenas a preocupação em criar novas dependências entre tarefas baseadas no fluxo de controle. Não existem meios para tratar o fluxo de dados nesta abordagem, não sendo possível prover troca de resultados intermediários. Além do que, criar relacionamentos declarativos insere muita complexidade à modelagem do processo, pois exige que os relacionamentos sejam criados antes de seu uso.

Na área de modelagem de processos de software a linguagem PROMENADE [14] oferece um meio de formalizar os modelos de processos de software, visando aumentar a expressividade, padronização, flexibilidade e reuso dos mesmos. A linguagem define quatro tipos de relacionamento de precedência básicos como um conjunto completo para a modelagem de processos de software básicos. Estes tipos de relacionamento são definidos de maneira declarativa, e podem ser combinadas a fim de criar-se relações de precedência derivadas. Mais uma vez os problemas são: a inexistência de meios para tratar o fluxo de dados e o acréscimo de complexidade na modelagem com o uso destes novos relacionamentos.

Joeris e Herzog [7], [8] defendem que a definição correta do comportamento das tarefas é a base para modelagem de workflow menos restritivos e para suporte a alterações dinâmicas. Neste sentido é apresentada uma maneira de especificar diferentes tipos de dependência no fluxo de controle, definidos basicamente por regras do tipo E-C-A (Evento-Condição-Ação). O mecanismo de disparo trata as tarefas como se estas fossem componentes reativos, que encapsulam seu comportamento interno e interagem com outras tarefas através de passagem de mensagens/eventos. As dependências de fluxo de controle são refinadas de maneira a apoiar processos heterogêneos e flexíveis. Processos cooperativos são suportados neste sistema através da possibilidade de integrar versões à semântica do modelo de workflow no nível conceitual.Assim como nas propostas anteriores, o grande problema relacionado a esta abordagem está na necessidade de criação e utilização de novos relacionamentos quando da modelagem dos processos. Isso insere muita complexidade extra ao modelo.

2.2 Flexibilidade *a posteriori*

Uma das propostas de flexibilidade a posteriori é a abordagem de Nickerson [11], que apresenta o problema que as transações longas podem causar. Para tal é proposto um modelo de workflow baseado em eventos, através do qual é possível alterar o andamento de uma tarefa longa. A solução apresentada por Nickerson flexibiliza a restrição de isolamento das tarefas, através de eventos externos capazes de interromper uma tarefa.

No sistema ADEPT [13] é proposto um suporte a mudanças no modelo quando da instanciação dos processos. O ADEPT é um modelo conceitual de workflow baseado em grafos que possui uma base formal para sua sintaxe e semântica. Baseado neste modelo foi criado um conjunto completo e mínimo de operações de modificação que apóia os usuários na modificação da estrutura de instâncias de processos de workflow em execução, preservando sua corretude e consistência – ADEPTflex [12]. Tais verificações dizem respeito tanto ao fluxo de dados quando ao fluxo de controle. O ADEPTflex compreende operações para inserção, remoção e alteração na ordem das tarefas. Alterações dinâmicas no modelo em tempo de execução podem ser de grande utilidade em processos com características intelectuais e cooperativas. Porém, tal abordagem esbarra na mesma rigidez encontrada nos sistemas tradicionais, pois as tarefas continuarão sendo caixas pretas para os usuários, durante a execução.

O projeto Co-flow [9] traz uma abordagem de suporte a workflow inteligentes, construído sobre o TriGSflow. Nesta abordagem, agentes inteligentes são habilitados a adaptar e estender os modelos de processos pré planejados para suas situações específicas, sem influenciar os outros casos baseados no mesmo modelo. A adaptação dos modelos de processo pode ser iniciada pelos agentes e as possibilidades de adaptação podem ser apoiadas por um sistema de recomendação, mostrando aos agentes as operações possíveis no contexto corrente. Mais um trabalho que apresenta solução para o problema da alteração dinâmica de processos. É possível utilizar este tipo de abordagem como suporte à antecipação, porém, como dito anteriormente, isto demandaria muito custo e as tarefas continuariam como caixas pretas, executando em isolado.

Coo-Flow [5], é outra abordagem que apresenta flexibilidade *a posteriori*, mais especificamente é neste trabalho que é proposta a antecipação de tarefas. Para permitir que as tarefas fossem disparadas antes da conclusão de suas antecessoras, passando assim para os estados propostos foi necessária a criação de novos estados para as tarefas e modificação do algoritmo de execução. Esta proposta serviu como motivação inicial deste trabalho, sendo a base de

toda a pesquisa aqui realizada. Esta abordagem apresenta a possibilidade de levar à fase de execução dos processos a antecipação das tarefas.

3. ANTECIPAÇÃO DE TAREFAS

Segundo Grigori, Charoy e Godart [5] antecipação é um conceito que, intuitivamente, significa que uma tarefa pode iniciar sua execução mesmo que antes do previsto. Mesmo sabendo que é possível antecipar o início de uma tarefa em decorrência da conclusão de suas antecessoras, antes do prazo previsto, não é esta a antecipação tratada neste trabalho. O conceito de antecipação aqui utilizado é iniciar uma tarefa mesmo que suas antecessoras diretas não tenham sido concluídas. Evidentemente, este início acontece durante a execução das tarefas antecessoras. Também, a conclusão das tarefas antecipadas depende da conclusão destas antecessoras.

A antecipação implica no aumento da cooperação entre tarefas e em melhor desempenho com relação ao tempo de execução dos processos. O aumento na cooperação está relacionado diretamente à troca de resultados, permitindo que duas ou mais tarefas trabalhem sobre instâncias de um determinado dado que ainda está sendo produzido.

Mas, a grande vantagem de utilizar este tipo de antecipação é o aumento na cooperação entre duas tarefas e a diminuição do tempo total de execução do processo. Isto se deve ao paralelismo (parcial) criado entre tarefas que deveriam executar de maneira estritamente seqüencial. Tal acréscimo no paralelismo é ilustrado na Figura 1. A Figura 1(a) representa o fluxo tradicional que o processo deveria seguir, condicionando o inicio de uma tarefa ao final de sua antecessora. A Figura 1(b) apresenta o fluxo do mesmo processo com a antecipação de uma tarefa (tarefa Revisão). Nesta última figura o gatilho da antecipação é o fornecimento de um resultado intermediário. A conclusão da tarefa Revisão tem como condição para conclusão o envio do resultado final da tarefa Escrita. No exemplo apresentado fica evidente o ganho com relação ao tempo de execução do processo, ao aumentar o paralelismo.

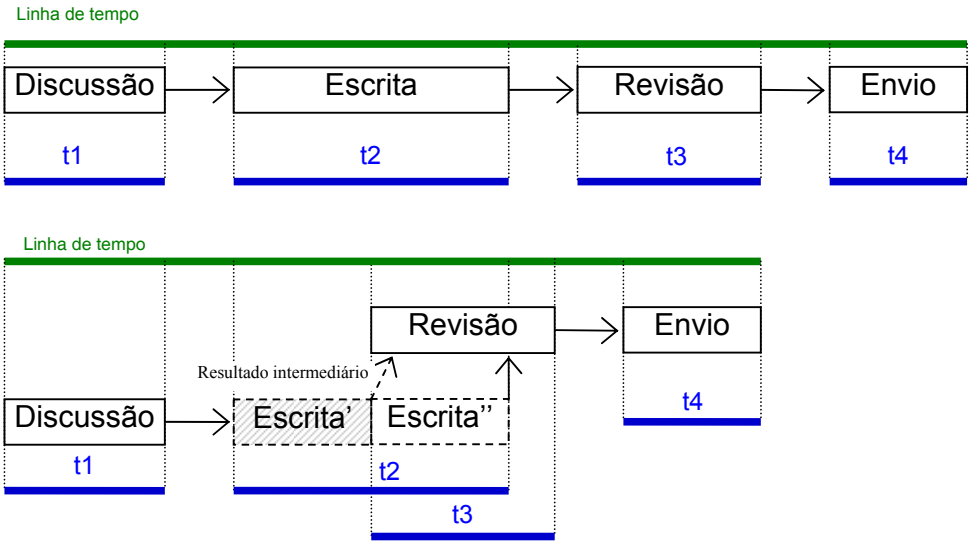


Figura 1. Aumentando o paralelismo através da utilização de antecipação

Em SGWfs tradicionais os processos de workflow são modelados de forma que as tarefas sejam caixas pretas e obedeçam a uma dependência do tipo fim-início. De certa forma, isto quer dizer que, nos SGWfs tradicionais, uma tarefa:

- é atômica;
- deve ser executada de maneira isolada;
- pode ser somente iniciada quando suas antecessoras forem concluídas; e
- precisa que seus dados de entrada (quando houverem) sejam resultados finais de suas tarefas antecessoras.

A fim de permitir a flexibilização de alguns destes pontos foi proposta a antecipação de tarefas [4]. Com isto as tarefas não mais são vistas como caixas pretas, havendo a possibilidade de troca de resultados.

Na próxima seção são apresentadas as duas possiíveis maneiras de gerenciar a antecipação de tarefas: *a priori* e *a posteriori*.

4. MECANISMOS DE ANTECIPAÇÃO DE TAREFAS

A presente seção apresenta os dois mecanismos de gerenciamento de antecipação possíveis: antecipação gerenciada *a priori* e antecipação gerenciada *a posteriori.*

A antecipação gerenciada a priori prevê alterações no processo em tempo de definição, sem a necessidade de qualquer alteração na máquina de workflow. Esta abordagem exige esforço extra durante a modelagem, mas pode ser utilizada em qualquer SGWf existente atualmente [15].

A antecipação gerenciada a posteriori prevê alterações na máquina de execução do SGWf, de maneira a acabar com as restrições de isolamento e atomicidade das tarefas. Para tal, serão utilizados como base os novos estados propostos por Grigori [4] e o diagrama de estados proposto pela WfMC [20]. Para enriquecer o modelo, nesta dissertação é identificada a necessidade e é estabelecido um meio de identificar, em tempo de definição dos processos, as tarefas antecipáveis e o mapeamento desta identificação para a máquina de workflow. Também são apresentadas diferentes maneiras de lidar com dependências de controle e com o fluxo de dados e o mapeamento destas dependências para o diagrama de estados das instâncias de tarefas.

4.1 Antecipação Gerenciada *a priori*

Como dito anteriormente a antecipação é um conceito intuitivo, assim como a possibilidade de prevê-la durante a modelagem do processo. No exemplo ilustrado na Figura 1 fica claro que é possível utilizar os SGWfs tradicionais para conseguir o ganho de paralelismo oferecido pela antecipação de tarefas, apenas modificando o modelo de definição de processos. Este tipo de antecipação será chamado aqui de antecipação gerenciada a priori, pois ocorre em tempo de definição do processo de workflow, antes da execução.

A maneira de possibilitar a antecipação gerenciada a priori permitindo que haja troca de resultados intermediários é modelar as tarefas que podem ser antecipadas numa granularidade menor. Não se trata de uma nova abordagem, mas sim de uma nova prática de modelagem utilizando o modelo tradicional, sem a necessidade de novos elementos de modelagem ou novos conceitos.

Nesta nova prática de modelagem as tarefas antecipáveis e suas antecessoras devem ser "quebradas" em tarefas menores, ficando estas em parte seqüenciais e, em parte paralelas entre si. A Figura 2 ilustra como ficaria o modelo para o processo de concepção do artigo científico prevendo a antecipação de uma tarefa já durante a modelagem.

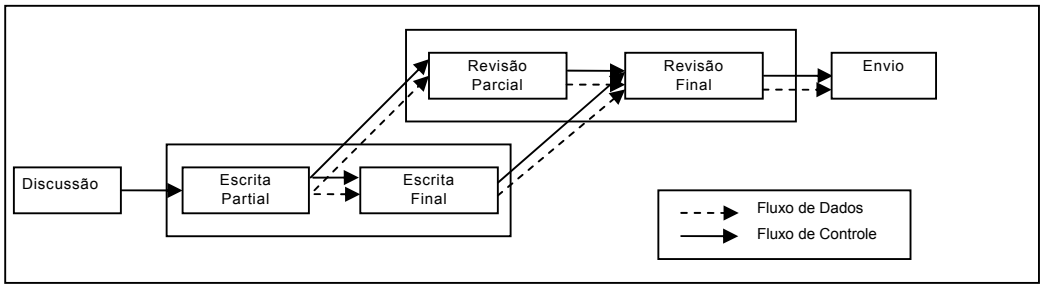


Figura 2. Antecipação utilizando modelo tradicional de definição de processos utilizando menor granularidade na modelagem das tarefas

Como pode ser visto, através da modelagem das tarefas em outro nível de granularidade é possível pré-determinar que tarefas sejam antecipadas. Neste caso o comportamento é o mesmo apresentado na Figura 1, deixando que as tarefas de escrita e revisão sejam executadas parcialmente em paralelo. Não é necessário utilizar nenhum tipo de elemento ou conceito adicional para este tipo de modelagem. Também não se faz necessário qualquer alteração na máquina de workflow. Qualquer ferramenta de definição de processos é capaz de modelar tal situação, e qualquer máquina de workflow pode interpretar, executar e gerenciar o referido processo.

O gerenciamento da antecipação a priori pode ser provido de duas maneiras distintas:

- Manual: Os workflow designers são responsáveis por identificar e modelar as tarefas já na granularidade com que estas devem ser instanciadas e executadas. Uma tarefa que pode ser antecipada deve ter suas antecessoras divididas em "sub-tarefas", prevendo a maneira que deve se dar a antecipação e quantos resultados intermediários (se houverem) serão enviados para as tarefas antecipáveis.
- Automática: Ao modelar um processo que possui tarefas que possam ser antecipadas, estas devem ser apenas identificadas pelo workflow designer responsável. Quando da instanciação deste processo, ele passará por um mecanismo de transformação que irá "quebrar" as tarefas identificadas, gerando-as numa granularidade maior. Tal granularidade poderá ser definida no momento da modelagem (através de um

atributo extra na tarefa), ou no momento da transformação do modelo (através de um parâmetro passados ao mecanismo de transformação). No caso da definição do processo estar representada em XPDL, pode-se adotar um script XSLT precedendo a máquina de workflow que ficaria responsável por gerar o uma nova definição em XPDL. Esta nova definição conteria todas as tarefas antecipáveis já na nova granularidade especificada.

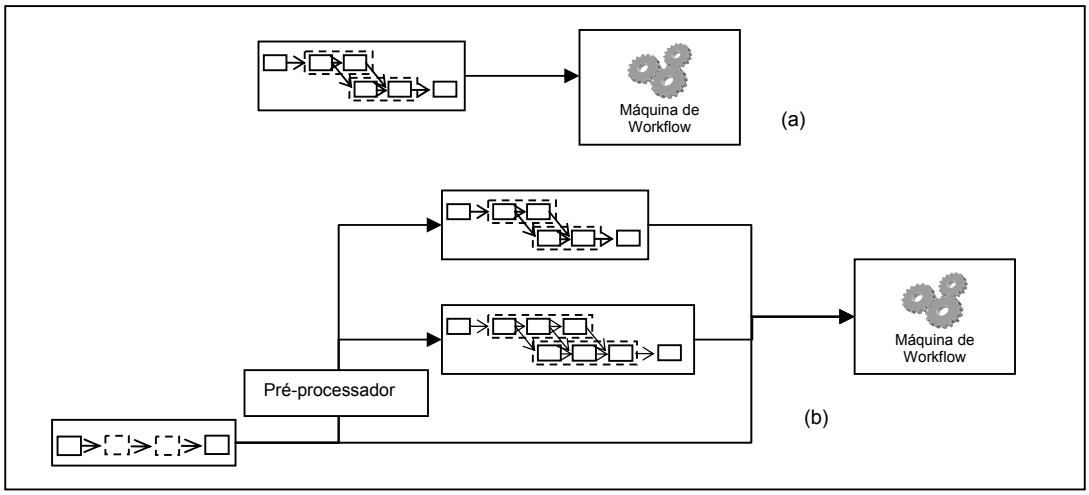


Figura 3. Antecipação gerenciada *a priori*: abordagem manual (a) e abordagem automática (b).

Quando tarefas antecipáveis são definidas manualmente não é necessário nenhum tipo de alteração no modelo tradicional de definição de processos. Apesar de possuir esta grande vantagem, é exigido muito esforço quando da modelagem dos processos.

Quando se utiliza a abordagem a priori automática, o esforço necessário é menor que usando a abordagem manual. Porém, é necessário que se modifique o modelo de definição de processos através da inserção de um identificador de tarefas antecipáveis. Uma vantagem de usar o mecanismo automático é a possibilidade de utilizar a definição de processos sem a mudança da granularidade. Porém, esta abordagem exige esforço de implementação de um pré-processador que faça a transformação do processo.

A Figura 3 ilustra a maneira como a antecipação *a priori* funciona em ambos os casos: gerenciamento manual (Figura 3(a)) e gerenciamento automático (Figura 3(b)).

4.2 Antecipação Gerenciada *a posteriori*

Tendo em vista a dificuldade de prever com precisão a antecipação em tempo de definição de processos, são apresentadas alterações na máquina de workflow que a permitam gerenciar a antecipação em tempo de execução. Neste trabalho este tipo de abordagem é chamada a posteriori.

Esta abordagem foi construída com base na proposta por Grigori [4], que prevê alteração nos estados das tarefas para permitir a antecipação. Porém, aqui são explicitados meios para identificar as tarefas antecipáveis e como isto é mapeado para a máquina de workflow. Outra contribuição ao modelo é a maneira com que são tratados as dependências de dados e controle e o mapeamento destas para o diagrama de estados das instâncias de tarefas.

4.2.1 Identificando as tarefas

A antecipação não é aplicável a todos os processos nem a todas as tarefas de um determinado processo. Existem processos e tarefas que devem ser executadas obrigatoriamente de maneira atômica e isolada. Algumas tarefas não executadas por recursos humanos (executadas automaticamente), possivelmente não saberiam como lidar com um resultado intermediário. Outro grupo de tarefas que não podem ser antecipadas são aquelas tarefas que necessitam de um resultado obrigatoriamente final para serem executadas. É o caso da tarefa de envio de um artigo, que não pode ser realizada em partes, através do envio de resultados parciais.

Visto isto, se torna necessário adicionar uma informação às tarefas indicando se uma tarefa pode ou não ser antecipada. Contudo, dizer que uma tarefa será antecipada já na fase de modelagem é um tanto quanto complexo. Porém, dizer quais tarefas podem ou não podem ser antecipadas é possível (e desejável). Com isto, ficariam identificadas para o sistema aquelas tarefas que possuem permissão para executar de maneira antecipada e aquelas que devem seguir sua execução de modo tradicional, necessitando aguardar todas suas condições de disparo serem satisfeitas.

A necessidade de tal identificação, porém, esbarra em alguns problemas apontados em outras abordagens, que são o aumento da complexidade na definição das tarefas e a alteração do modelo de processos tradicional. A alternativa encontrada foi a adição de um atributo às tarefas. Tal atributo é de caráter não obrigatório, e deixaria a critério do gerente responsável pela criação do processo a decisão de utilizar a antecipação ou não.

4.2.2 Modificando a máquina de execução

Para permitir que tarefas possam ser disparadas sem que suas condições sejam satisfeitas foi necessário alterar a máquina de workflow. É necessário que a máquina de execução possa identificar o novo atributo de identificação das tarefas e possa lidar com tipos diferentes de dependências de fluxo de controle e de fluxo de dados.

A fim de prover a flexibilidade quando da execução do processo, foi necessário, então, alterar o diagrama de estados das tarefas da máquina de execução do *SGWf*. Tal alteração é proposta a fim de tornar a antecipação clara e simples. Para a concepção de tal diagrama, foram tomados como base o diagrama de transição de estados proposto pela WfMC [20] e os estados e comportamento previstos no *Coo-Flow* [5], que leva em conta a antecipação de tarefas. O diagrama de transição e seus respectivos estados propostos são apresentados na Figura 4.

Neste novo diagrama foram criados os novos estados `pronta` e `nãoPronta` (sub-estados de `nãoRodando`) e os estados `antecipando` e `executando` (sub-estados de `rodando`). O estado `suspensa` foi re-alocado como um sub-estado de `nãoRodando`, uma vez que, quando uma instância de tarefa está `suspensa`, obviamente, não está rodando. Em níveis de granulosidade mais baixa, foram definidos alguns outros estados. O estado `pronta` é dividido ainda em dois sub-estados: o estado `paraAntecipar` e `paraExecutar`. O estado `nãoPronta` faz distinção entre os sub-estados `antecipável` e `nãoAntecipável`.

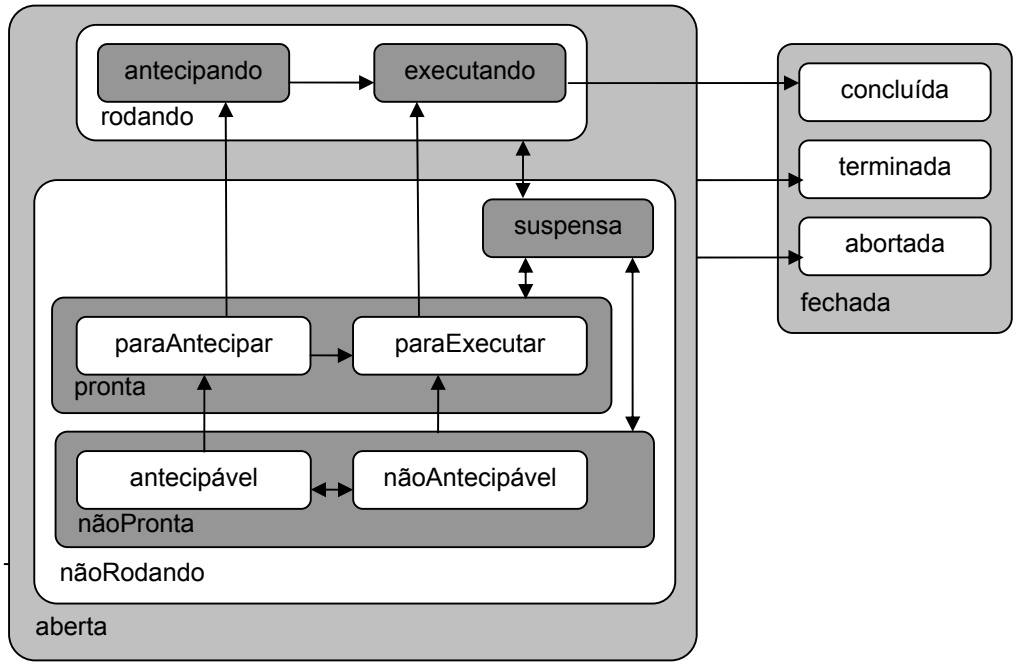


Figura 4. Diagrama de Transição de Estados para as tarefas

A seguir são apresentados os estados propostos e a descrição de cada um:

- `suspensa`: continua com o mesmo significado, indicando que uma instância de tarefa foi parada, porém pode ser retomada a qualquer instante;
- `nãoPronta`: é o estado inicial das instâncias. Indica que a tarefa foi instanciada, porém ainda não tem suas condições satisfeitas para ser disponibilizada para execução;
    - `antecipável`: indica que a instância da tarefa poderá ser antecipada se tiver suas condições de antecipação satisfeitas;
    - `nãoAntecipável`: indica que a instância da tarefa não poderá ser executada antecipadamente;

- `pronta`: a instância teve suas pré-condições (de execução ou antecipação) satisfeitas e está pronta para ser iniciada;
- `paraAntecipar`: indica que a instância da tarefa teve as pré-condições de antecipação satisfeitas e aguarda ser iniciada;
    - `paraExecutar`: a instância pode ter sua execução iniciada, pois suas pré-condições de execução foram satisfeitas por completo;
    - `antecipando`: tarefa está sendo rodada de maneira antecipada;
    - `executando`: tarefa está executando e pode ser concluída normalmente.

Quando se instancia um processo, todas suas tarefas são instanciadas. Quando instanciadas as tarefas são levadas para seu estado inicial. Neste caso o estado inicial das instâncias de tarefa é o estado `nãoPronta`. A instância é então mapeada para um dos dois sub-estados: `antecipável` ou `nãoAntecipável`. Tal classificação é realizada durante a fase de modelagem do processo (como descrito na seção anterior). A transição bidirecional existente entre estes dois estados se deve à possibilidade de alteração desta classificação mesmo durante a execução do processo. Enquanto a instância da tarefa não passou ao estado pronta é possível que o gerente modifique sua condição passando-a de `antecipável` a `nãoAntecipável` e vice-versa. Possibilitando a mudança de estado durante a execução aumenta-se a flexibilidade do sistema, deixando inclusive que agentes solicitem a antecipação de tarefas modeladas como não antecipáveis. Além disso, é possível adequar cada instância de um processo para suas necessidades e características próprias apenas alterando o estado de uma tarefa, sem que seja preciso modificar o modelo do processo, ou evoluir o esquema do *workflow*. A complexidade inserida por tal transição é mínima se comparada à complexidade necessária para evoluir o esquema do processo para cada instância em tempo de execução.

A seguir os comportamentos das tarefas antecipáveis e não antecipáveis serão explicados, através do detalhamento das transições entre os estados.

Primeiramente serão atacadas as tarefas que não podem utilizar a antecipação em seu andamento, isto é, as tarefas `naoAntecipaveis`. Uma tarefa `naoAntecipavel` terá seu funcionamento similar ao funcionamento das tarefas em sistemas de *workflow* tradicionais.

Sendo assim, uma tarefa `naoAntecipavel` se torna `pronta.paraExecutar` assim que todas suas condições de disparo forem satisfeitas (suas antecessoras forem `concluídas` e os dados de entrada forem resultados *finais*). A exceção são as tarefas iniciais do processo, que passam ao estado `pronta.paraExecutar` quando o processo é instanciado. Neste estado a tarefa fica disponível na lista de trabalho (*worklist*) dos agentes capazes de executá-las.

Para que a tarefa passe então do estado `pronta.paraExecutar` para o estado `executando` é necessário que esta seja escolhida por algum dos agentes para a qual esta foi oferecida e este comece a executá-la. Ao passar para tal estado a tarefa pode fornecer resultados (*intermediários* e/ou *finais*) e pode ser `concluída`, quando suas pós-condições forem satisfeitas.

Além desta transição, uma tarefa `executando` pode passar ao estado `suspensa` a qualquer instante. Este estado se refere às tarefas que foram paralisadas por escolha do agente responsável (ou do gerente), podendo esta tarefa ser retomada a qualquer instante. Quando retomada, a tarefa volta ao estado `executando`, podendo ser `concluída` ou `suspensa` novamente.

Já uma tarefa classificada como `antecipável` pode ter seu comportamento modificado justamente pela flexibilidade oferecida pela antecipação. O fato de uma tarefa ser dita `antecipável` não garante que esta será antecipada, apenas permite que esta possa ser antecipada. É apenas uma alternativa para possibilitar a cooperação entre os agentes (através da troca de resultados *intermediários*) e ajudar na redução do tempo de execução dos processos.

Uma tarefa `antecipável` pode seguir a trajetória de uma tarefa `naoAntecipavel`. Isto é, quando suas pré-condições forem satisfeitas passa ao estado `pronta.paraExecutar`, ficando disponível na lista de tarefas dos agentes capazes de executá-la. Ao ser escolhida por um agente, passa a estar `executando`, podendo então ser `suspensa` ou `concluída`.

O comportamento de uma tarefa `antecipável`, porém, pode ser totalmente distinto daquele de uma tarefa `nãoAntecipável`. Quando uma tarefa `antecipável` tem suas pré condições de antecipação (definidas quando da modelagem) satisfeitas, esta passa ao estado `pronta.paraAntecipar`. Este estado funciona de maneira similar ao estado `pronta.paraExecutar`. Quando `pronta.paraAntecipar`, a tarefa aparece na lista de tarefa dos agentes capazes de executá-la. A partir deste estado é possível que a tarefa se torne `executável` ou passe a estar `antecipando`.

A transição entre o estado `pronta.paraAntecipar` e `executável` é ativada quando as pré condições de disparo da tarefa são satisfeitas antes desta iniciar sua antecipação. Neste caso, a tarefa volta a seguir o fluxo normal, aparecendo como `pronta.paraExecutar` na lista de tarefas dos usuários.

Já a transição entre `pronta.paraAntecipar` e `antecipando` é ativada quando algum dos agentes capazes de executar a tarefa selecionam-na para antecipação. A partir deste momento a tarefa está `antecipando`. Neste estado o comportamento é muito parecido com uma tarefa `executando`, estando a diferença no fato das tarefas não poderem ser `concluídas` enquanto estão `antecipando`. Para poderem ser `concluída` uma tarefa necessita primeiramente passar pelo estado `executando`. Esta restrição se deve à necessidade de garantir que a tarefa está obedecendo às suas pré-condições de execução (e não de antecipação).

5. CONCLUSÕES

Este trabalho apresenta duas abordagens de antecipação de tarefas que permitem maior flexibilidade nos SGWfs tradicionais. Como vantagem direta do uso abordagens está o aumento no espectro de aplicações que podem ter seus processos gerenciados por SGWfs. Além disso, uma das abordagens apresenta como vantagem o aumento da cooperação entre tarefas, que pode conduzir a produtos finais de melhor qualidade.

As duas abordagens identificadas e apresentadas são classificadas de acordo com o momento em que a antecipação é gerenciada (tempo de definição ou tempo de execução), sendo chamadas a priori e a posteriori.

A abordagem a priori apresenta a antecipação gerenciada em tempo de definição dos processos. Ela é estabelecida a fim de tornar a antecipação uma prática possível em qualquer SGWf, apenas propondo uma maneira diferente de modelar as tarefas. É proposta uma nova prática de modelagem em que, apenas alterando a granularidade das tarefas envolvidas na antecipação torna-se possível antecipar as tarefas. Apesar de contar com a vantagem de poder utilizar qualquer SGWf existente, a flexibilidade provida por esta abordagem é pequena. Esta flexibilidade se limita ao ponto em que são realizados os envios de resultados intermediários, sendo o número de troca de resultados pré-definido. Então, pode-se dizer que, neste caso, a flexibilidade oferecida é pré-determinada, uma vez que as tarefas continuam sendo atômicas.

A abordagem *a posteriori* aumenta a flexibilidade oferecida, fazendo o gerenciamento a posteriori da antecipação, isto é, em tempo de execução dos processos. A abordagem deste tipo apresentada neste capítulo tem base no diagrama de estados proposto por Grigori [4] e define meios de antecipar tarefas através da flexibilização dos fluxos de controle e dados.

A flexibilidade oferecida por este tipo de antecipação é muito maior se comparada à abordagem a priori, não obrigando a antecipação e não definindo o número de interações que deve haver durante a execução do processo. Além disso, nesta abordagem o esforço necessário durante a modelagem é menor que na abordagem a priori. A desvantagem da utilização da antecipação gerenciada a posteriori fica por conta do esforço necessário para alterar da máquina de workflow.

Tabela1. Comparativo entre os mecanismos de gerenciamento apresentados

	Gerenciamento a priori		***Gerenciamento a posteriori***
	Manual	*Automático*	
Novos elementos de modelagem	Nenhum	Necessita atributo de identificação de tarefas	Atributo de identificação
Máquina de Workflow	Qualquer		Específica para suportar antecipação
Implementação extra	Nenhuma	Pré-processador	Sim
Reuso de definição de processo	Uma definição para cada caso	Uma definição para vários casos: alterando parâmetros	Sim. Antecipação é de total responsabilidade da máquina.
Gasto com modelagem	Alto. Necessita prever a antecipação		Pequeno. Definições podem ser realizadas na execução
Flexibilidade	Pré-definida. Execução continua rígida		Alta

Como resultado da pesquisa obteve-se a tabela 1, que traz um comparativo levando em conta as duas possíveis maneiras de prover antecipação *a priori* e a antecipação *a posteriori*. Na tabela as células sombreadas representam a melhor alternativa com relação ao item especificado na linha correspondente.

Alguns experimentos estão sendo realizados levando-se em conta a antecipação de tarefas, tentando prever o comportamento humano durante tal processo. Com certeza o uso desta abordagem trará perdas com relação ao esforço de comunicação relativo à troca dos resultados intermediários. O que se pretende ter como resultado de tais experimentos, é a relação existente entre antecipação e perda com comunicação, podendo-se prever os casos em que a antecipação traria ganhos com relação ao tempo de execução dos processos.

Protótipos que suportem estão sendo implementados como parte do projeto CEMT [10], sendo uma delas a alteração na ferramenta de definição de processos Amaya Workflow [18], [19] e a concepção de uma ferramenta de simulação [17] para permitir a realização dos experimentos.

Agradecimentos

Agradecimentos especiais ao Programa de Pós Graduação em Computação da Universidade Federal do Rio Grande do Sul, ao grupo CEMT/SIGHA e ao CNPq, que forneceram suporte físico e financeiro a esta pesquisa. Agradecemos ainda ao apoio das instituições de origem dos autores: Universidade Estadual de Maringá, Cesumar e Universidade Federal do Rio Grande do Sul.

Referências

[1] AALST, W.; HEE, K. Workflow Management: Models, Methods and Systems. The MIT Press – Massachussets Institute of Technology, Massachussets, 2002.

[2] DOURISH, P.; HOLMES, J.; MACLEAN, A.; MARQVARDSEN, P.; ZBYSLAW, A. Freeflow: Mediating Between Representation and Action in Workflow Systems. In: ACM conference on computer supported cooperative work, 1996, Boston, EUA. Proceedings… Nova York: ACM Press, 1996. p. 190-198.

[3] GODART, C.; PERRIN, O.; SKAF, H. Coo: a Workflow Operator to Improve the Cooperation Modeling in Virtual Processes. In: International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises, 9., 1999, Sidney, Austrália. Proceedings… Washington: IEEE Computer Society, 1999. p. 126-131.

[4] GRIGORI, D. Eléments de flexibilité des systèmes de workflow pour la définition et l'exécution de procédés coopératifs. 2001. 135f. Tese (Doutorado em Informática) – Université Henri Poincaré, Nancy 1. 2001.

[5] GRIGORI, D.; CHAROY, F.; GODART, C. Coo-Flow: A Process Technology to Support Cooperative Processes. International Journal of Software Engineering and Knowledge Engineering, v. 14, n.1, p. 1-19, Janeiro 2004.

[6] HAGEN, C.; ALONSO, G.. Beyond the black box: Event-based inter-process communication in process support systems. In: International Conference on Distributed Computing Systems,19. , Austin, EUA, 1999. Proceedings…IEEE Computer Society, 1999. p. 450–457.

[7] JOERIS, G.; HERZOG, O. Towards Object-Oriented Modeling and Enacting of Processes. TZI-Report 07/98, Center for Computing Technologies, Universidade de Bremen, Bremen. 1998.

[8] JOERIS, G. Defining Flexible Workflow Execution Behaviors. In: Enterprise-Wide and Crossenterprise Workflow Management: Concepts, Systems, Applications, 1999, Ulm, Alemanha. Proceedings… 1999.

[9] KRAMLER, G.; RETSCHITZEGGER, W. Towards Intelligent Support of Workflow. In: Americas Conference on Information Systems (AMCIS), 2000, Long Beach, EUA. Proceedings… 2000. p.581-585.

[10] LIMA, J. V.; QUINT, V.; LAYAIDA, N.; EDELWEISS, N.; ZEVE, C. M. D.; PINHEIRO, M. K.; TELECKEN, T. L. The Conception of Cooperative Environment for Editing Multimedia Documents with Workflow Technology (CEMT). In: PROTEM-CC, 4., 2001, Rio de Janeiro. Projects Evaluation Workshop: international cooperation: proceedings. Brasília: CNPQ, 2001. p. 542-560.

[11] NICKERSON, J.V. Event-based Workflow and the Management Interface. In: Hawaii International Conference on System Sciences (HICSS), 36., 2003, Big Island, Havaí. Proceedings… Washington: IEEE Computer Society, 2003.

[12] REICHERT, M.; DADAM, P. Adeptflex – Supporting Dynamic Changes of Workflow Without Loosing Control. Journal of Intelligent Information System: Special Issue on Workflow Management Systems, v.10, n.2, p. 93-129, março/abril 1998.

[13] REICHERT, M.; RINDERLE, S.; DADAM, P. ADEPT Workflow Management System: Flexible Support for Enterprise-Wide Business Processes. In: Business process management international conference (BPM), 2003, Eindhoven, Holanda. Proceedings… Berlin: Springer-Verlag, 2003a. p. 370-379.

[14] RIBÓ, J.M.; FRANCH, X. A Precedence-based Approach for Proactive Control in Software Process Modeling. In: international conference on software engineering and knowledge engineering (SEKE), 14., 2002, Ischia, Itália. Proceedings… Nova York: ACM Press, 2002. p. 457-464.

[15] STEINMACHER, I.; LIMA, J.V.; FREITAS, A.V.P. Uma Maneira de Antecipar Tarefas em um Sistema Gerenciador de Workflow In: Congreso Argentino de Ciencias de la Computación (CACIC), 2004, Buenos Aires, Argentina. Proceedings... 2004a.

[16] STEINMACHER, I. Estudo e Análise de Desempenho da Antecipação de Tarefas em Sistemas Gerenciadores de

Workflow. 2005, 108 f. Dissertação (Mestrado em Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre. 2005.

[17] STEINMACHER, I. Um Simulador para Avaliação da9 Antecipação de Tarefas em Sistemas Gerenciadores de Workflow. In: Forum Internacional de Software Livre - Workshop de Software Livre, 2005, Porto Alegre. Anais... 2005. *A ser publicado.*

[18] TELECKEN, T. Um Estudo sobre Modelos Conceituais para Ferramentas de Definição de Processos de Workflow. 2004, 118 f. Dissertação (Mestrado em Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre. 2004.

[19] W3C. Amaya Home Page. Disponível em: <http://www.w3.org/Amaya/>. Acesso em: 2004a.

[20] WFMC. Workflow Management Coalition Home Page. Disponível em: <http://www.wfmc.org>. Acesso em: 2004.

Patrones de Integración de Servicios Groupware

Federico Naso
LIFIA, Facultad de Informática, UNLP.
La Plata, Buenos Aires, Argentina.
federico.naso@lifia.info.unlp.edu.ar

and

Alejandro Fernandez
LIFIA, Facultad de Informática, UNLP
La Plata, Buenos Aires, Argentina
alejandro.fernandez@lifia.info.unlp.edu.ar

Abstract

The main intention of groupware applications is to provide support for groups of users which are working cooperatively. Groupware applications are defined by the one or more groupware services they provide [3]. Each time it is more common to find application with groupware services integrated. This integration of services is made integrating some common aspect using framework or protocols. Protocols are the business rules between its services. The design methodology using frameworks or protocols to integrate services is commonly oriented to the development. The lack of assistance in groupware services integration design, examples and architectures, entails that each integration has to be done completely from zero. This paper proposes a compilation of groupware services integrations patterns. This compilation contains recurrent groupware services integrations and offer solutions to this problem.

Keywords: Groupware services integration, groupware services, patterns**.**

Resumen

El propósito de las aplicaciones groupware es brindar soporte a grupos de usuarios que se encuentran trabajando cooperativamente. Las mismas son definidas según uno o varios servicios groupware que brindan [3]. Cada vez es más común ver aplicaciones donde se integran servicios groupware. Dicha integración de servicios groupware se consigue integrando aspectos que son "comunes" mediante la utilización de frameworks o mediante el uso de protocolos estándares que conforman las reglas de negocio entre los servicios. Las alternativas existentes en casos se ven limitadas por la lista de servicios que permiten integrar o por las tecnologías utilizadas para la implementación (p.e., lenguajes de programación). La falta de asistencia en el diseño de integraciones de servicios groupware, ejemplos y arquitecturas que sirvan de guía, conlleva a que cada vez que alguien necesite realizar una integración, esta se lleve a cabo desde cero. Este paper propone un catálogo de patrones de integración de servicios groupware. El catálogo captura casos recurrentes de integración de servicios groupware y brinda soluciones ya probadas a estos problemas.

Palabras claves: Integración de servicios groupware, servicios groupware, patrones.

1. Introducción

En Usability First [1] se define al groupware como: "la tecnología diseñada para facilitar el trabajo en grupo. Esta tecnología puede ser usada para comunicar, cooperar, coordinar, resolver problemas, competir o negociar. Mientras las tecnologías tradicionales tales como el teléfono son calificadas como *groupware*, el término es comúnmente usado para hacer referencia a una clase de tecnología específica relacionada con las redes de computadoras modernas. Tecnologías tales como el e-mail, newsgroups o el Chat".

Una definición más formal de groupware es dada por Ellis [2] y sus colegas como: "Aplicaciones de software diseñadas para brindar soporte a grupos de personas en el área de comunicación, colaboración sobre objetos e información compartida"

El propósito de las *aplicaciones groupware* es brindar soporte a grupos de usuarios que se encuentran trabajando cooperativamente. Las *aplicaciones groupware,* como por ejemplo un calendario grupal o una pizarra compartida, son definidas según los *servicios groupware* que éstas brindan [3]. Una *aplicación groupware* esta compuesta por uno o varios *servicios groupware*. Un *servicio* es un conjunto de posibles interacciones entre un usuario y un sistema, donde dicho sistema, es el encargado de brindar soporte para tal interacción [4]. Subsecuentemente los *servicios groupware* permiten a los usuarios cooperar, comunicarse y colaborar utilizando información compartida. El mail, el Chat o la mensajeria instantánea son *servicios groupware*.

Los ejemplos de *aplicaciones groupware* que se dan a continuación, nos ayudan a tener una idea mas clara del concepto de *aplicación groupware* y *servicios groupware.*

> **Calendario Grupal:** permite la coordinación de reuniones entre personas de un grupo de trabajo. Los mismos se encargan de concretar reuniones teniendo en cuenta el tiempo libre de los participantes. Esta aplicación groupware brinda varios *servicios groupware;* el calendario compartido en donde se mantiene la información de las reuniones; un servicio de notificación, como por ejemplo el mail para notificar sobre las reuniones; un servicio de coordinación que puede ser una herramienta de votación para que los participantes puedan votar entre las diferentes alternativas a una reunión.
>
> **Pizarra Compartida:** permite a varias personas dibujar sobre un área compartida de trabajo. Las personas están comúnmente distribuidas en diferentes lugares geográficos y se comunican mediante el chat provisto por la misma aplicación. Con este ejemplo vemos que las pizarras compartidas proveen dos servicios groupware: la pizarra como un área de dibujo compartida; el chat como herramienta de comunicación sincrónica.

2. Integración de servicios Groupware

Cada vez es más común ver aplicaciones donde se integran servicios groupware. Ya en 1997 Roseman y sus colegas [5] hablaban de las expectativas que se tenía sobre la integración de "herramientas" en ambientes groupware como aplicaciones únicas. Un ejemplo de esto son las pizarras compartidas de la sección anterior que integran chat y dibujo colaborativo en una misma aplicación groupware. En la Ilustración 1 podemos ver el esquema de una aplicación groupware, en particular una pizarra compartida compuesta por servicios groupware (chat y dibujo).

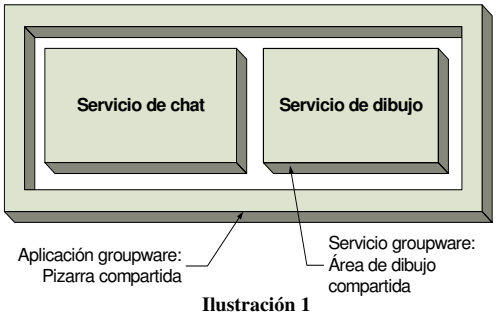


Ilustración 1

En la actualidad, en lugar de diseñar una solución groupware en función de los servicios individuales que brinda, se la diseña como un ambiente integral que ofrece una plataforma de colaboración a grupos de trabajo que se encuentran distribuidos remotamente y llevan a cabo interacciones electrónicas.

Podemos definir la *integración de servicios groupware* como "la actividad de seleccionar y combinar servicios groupware para obtener un nuevo tipo de servicio que aporte nueva funcionalidad, potenciando los servicios básicos que lo componen". Por ejemplo, teniendo en cuenta que "el todo es mas que la suma de las partes", una aplicación de calendario compartido a la que se le agrega un servicio de comunicación para la notificación de eventos como el e-mail y un servicio que permita coordinar las reuniones como un

servicio de votación, es una aplicación groupware mucho más funcional que el calendario por si solo, dado que brinda mayor soporte a la hora de tener que programar reuniones de grupos.

La *integración de servicios groupware* consigue haciendo que ciertos aspectos que son "comunes" a los servicios groupware se relacionen fuertemente en algunos de los siguientes aspectos que operan como punto de unión en la integración:

- Manejo de usuario: los usuarios deben tener una identidad única y ser solo una instancia en toda la aplicación, sin importar en que servicios sean utilizados.

- Awarenes o percepción: el awarenes posibilita conocer información sobre las acciones que están llevando a cabo otras personas en el área de trabajo [6]. Por ejemplo nos permiten saber quien se encuentra trabajando con un documento en particular y que hace o quien esta disponible para colaborar en ese instante.

- Sesiones: las sesiones son invocaciones a los sistemas groupware. Una sesión provee a un grupo de participantes, una interfaz a un contexto compartido [7].

- Datos: los datos son los objetos a compartir. Por ejemplo un documento, el dibujo de una pizarra compartida, los datos de un usuario.

- Protocolos: los protocolos son un conjunto de reglas formales, que ejemplo establecen el formato de los datos a utilizar entre los servicios, los estados de la comunicación, las reglas de negociación, etc. Mediante ellos se puede, por ejemplo, establecer una comunicación sincrónica entre usuarios. Para ello los usuarios deben usar un servicio que implemente dicho protocolo de comunicación.

Existen en la actualidad herramientas o ambientes, tales como BSCW [8] o TeamWave Workspace [9], que proveen servicios groupware integrados. Por ejemplo, TeamWave Workspace es un ambiente de groupware que soporta una amplia variedad de actividades colaborativas. TeamWave integra dentro de un mismo ambiente *whiteboards* colaborativos (o pizarras compartidas), *chats*, *sticky notes*, y *calendarios compartidos*. El sistema puede ser usado tanto sincrónica como asincrónicamente. [5].

3. Como integrar servicios Groupware

Los servicios groupware son integrados actualmente de dos formas: mediante la utilización de frameworks o mediante el uso de protocolos estándares que conforman las reglas de negocio entre los servicios.

Un framework provee guías arquitecturales para descomponer un diseño en clases abstractas y definen sus responsabilidades y relaciones. Predefine parámetros de diseño, con lo cual el diseñador solo tiene que concentrase en aspectos específicos de su aplicación. Los desarrolladores adaptan el framework para una aplicación en particular subclasificando y compartiendo instancias de las clases de dicho framework.

La mayoría de los frameworks [10, 11, 12, 13] brindan un conjunto básico de servicios, como por ejemplo el manejo de usuarios, la persistencia de los datos, manejo de sesiones, abstracción en la comunicación entre los clientes y el servidor, etc., y luego proveen soporte para desarrollar componentes que utilicen estos servicios integrándose al framework. Esto, obviamente no es compatible con los servicios ya desarrollados sin la utilización del framework, ni con las tecnologías tan heterogéneas actuales, dado que cada framework propone su propia tecnología y forma de desarrollo.

Si bien estas integraciones son efectivas, para el propósito deseado y dentro de contextos específicos, es extremadamente difícil reubicarlas en otros contextos dado que están fuertemente atados a las decisiones tecnológicas del framewok. Esto es a causa de que los servicios existentes utilizan tecnologías muy heterogéneas. Por ejemplo, un servicio desarrollado en Smalltalk no puede ser integrado con un servicio desarrollado en JAVA. Los diseños deberían plantear soluciones abstractas y sin tener en cuenta la tecnología subyacente.

En el caso de los protocolos, estos se usan para relacionar fuertemente aplicaciones que no comparten una implementación de algunos de los aspectos de integración. Se trata de un conjunto de reglas formales que permiten establecer comunicación de forma no ambigua entre las entidades que los implementen. Dado que son conjuntos de reglas, para usar un protocolo se debe implementar el mismo de manera integra. Es

decir, si quisiéramos tener un cliente de mail que se comunique con otros clientes de mail, deberíamos implementar completamente el protocolo de envío y recepción de mail en el lenguaje que deseáramos. En este caso tanto lo que transmitimos como lo que enviamos, es decir el mail, debe respetar el estándar RFC822. [14].

A diferencia de los frameworks los protocolos son más abstractos dado que regulan, mediante reglas pactadas, el modo en que las aplicaciones debe comunicarse entre si. En los frameworks se parte de código ya escrito y se reutiliza dicho código.

Como podemos ver en los dos casos anteriores, tanto en el uso de frameworks como en el uso de protocolos, la metodología de diseño en la integración de servicios groupware está típicamente orientada a la implementación. Éste tipo de metodologías provoca que cada vez que se necesite una solución se codifique la misma. De este modo se tiene una mayor consideración a los problemas inherentes a la implementación que aspectos relacionados con los *servicios groupware* [3]. Esto trae aparejado que los diseñadores no tengan una vista general en el diseño de las funciones que el sistema debe proveer y sus relaciones.

La falta de asistencia en el diseño de integraciones de servicios groupware, ejemplos y arquitecturas que sirvan de guía y ayuden a realizar integraciones efectivas, conlleva a que cada vez que alguien necesite realizar una integración, la misma, se lleve a cabo desde cero. Con lo cual los problemas que ya fueron resueltos por otras personas previamente se vuelvan a experimentar una y otra vez.

La contribución de este paper es la compilación de un catálogo de patrones de integración de servicios groupware. Dicho catálogo captura casos recurrentes de integración de servicios groupware y brinda soluciones ya probadas a estos problemas.

4. Trabajos Relacionados

En la categoría de los frameworks, el trabajo de Robert Slagter es el más representativo. En su tesis doctoral [3], propone un modelo de composición de servicios groupware que permite seleccionar y componer *módulos groupware* según sea necesario; además de hacer más fácil realizar diseños de servicios adaptables. Según Slagter un servicio groupware esta formado por una composición de módulos de servicios groupware (Groupware Service Modules). Un módulo de servicios groupware esta formado por múltiples *elementos* del módulo del servicio groupware (Groupware Service Module Element), donde cada elemento puede ser por ejemplo un pedido de inicio de conferencia (starConf), el fin de la conferencia (endConf), unirse a una sesión (join), agregar una herramienta (addTool), obtener el rol (getRole), etc. En base a los módulos y a los elementos propone e implementa en .NET un modelo de referencia (llamado CooPS) que define las relaciones entre los diferentes módulos de servicios groupware. CooPS puede ser visto como una especificación de una capa de aplicación en un sistema tradicional. Si bien este enfoque brinda soporte para la integración de servicios (para el modelado y construcción de servicios integrados), se ve limitada la sola composición de los elementos modulares identificados por Slagter y a la plataforma de implementación CooPS para .NET.

En la categoría de integración por protocolos no se registran trabajos de aplicación genérica, sino que los ejemplos que pueden encontrase se limitan a la integración de aplicaciones groupware especificas. Un ejemplo muy difundido es el que apunta a integrar distintos servicios de mensajería instantánea. Esto es posible gracias a la publicación de los estándares de comunicación de los distintos proveedores de servicios de mensajería instantánea, algunos ejemplos son: ICQ con su protocolo OSCAR [26], Jabber.org, con su protocolo abierto llamado Jabber/XMPP [27], Yahoo [28] tiene un protocolo propietario y cerrado, IMPP [29] protocolo que trata de imponer otro estándar.

5. Patrones de diseño

La noción de *patrones de diseño* está basada en el trabajo del arquitecto Christopher Alexander, manifestado en dos libros (*A TímelesWay of building* [15] y *A Pattern Language* [16]). Christopher Alexander y sus colegas del área de diseño de edificios, en su libro "*A Pattern Language"*, dan la siguiente definición de patrón: "Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma".

Los patrones de diseño han sido adoptados eficientemente en el diseño de software. Gamma, Helm, Johnson y Vlissides transfieren el concepto de patrones arquitecturales de Alexander a la ingeniería de software donde publican una colección de patrones de diseño orientado a objetos [25].

Un patrón describe, con algún nivel de abstracción, una solución experta a un problema. Normalmente, un patrón está documentado en forma de una plantilla.

Los *patrones de diseño* pueden ser utilizados como una alternativa en la documentación de los diseños de integración. Los patrones contribuyen por medio de arquitecturas mínimas que luego pueden ser implementadas por medio de tecnologías especificas. A través de los patrones de integración de servicios logramos abstraernos de la tecnología a utilizar y hacer hincapié en los *servicios groupware* a integrar.

Los patrones han sido agrupados y organizados en catálogos. Los patrones no son entidades aisladas, sino que forman parte de un conjunto de patrones que se complementan [16].

Cada catálogo provee diferentes clasificaciones y descripciones. Algunos catálogos describen patrones de análisis mientras que otros patrones de diseño [17, 18, 19], patrones de interacción [20], patrones sobre groupware [21], HCI [22] o IU [23] incluyendo también patrones de un dominio particular o independientes del dominio.

Los catálogos de patrones se utilizan para resolver problemas en un determinado dominio en particular, en nuestro caso el dominio es la integración de servicios groupware. Hacen que la reutilización de los patrones sea más fácil, clara y ordenada. Sin ellos no se podría utilizar patrones en el diseño.

La ventaja principal de estos catálogos, es la recopilación de experiencia de integración de servicios para que las personas puedan reutilizar una solución ya probada y documentada de integración. Esto hace más fácil la integración ya que el catálogo provee técnicas y métodos para que la integración se lleve a cabo de manera eficiente y eficaz.

6. Catálogo de Patrones de Integración

Esta sección muestra uno de los patrones del catálogo el cual fue presentado y evaluado en un congreso especifico de patrones de diseño [24]. Dicho patrón se llama rendezvous y se presenta en su forma completa.

Luego se presentan el resto de los patrones que componen el catálogo mediante una breve descripción de cada uno. De ellos solo se tienen en cuenta, el nombre, la intención, el problema, las fuerzas, la solución y un ejemplo en donde dicho patrón fue utilizado.

Patrón – Rendezvous[1]

Intención: Informar a usuarios de un sistema de groupware asincrónico la posibilidad de colaborar con otras herramientas de manera sincrónica (groupware sincrónico).

También conocido como: Lugar de encuentro, Reunión, Confluencia.

Problema: La carencia de indicadores de presencia y la imposibilidad de iniciar colaboración sincrónica que son inherentes a los servicios de groupware asincrónico conducen, en casos donde los usuarios se encuentran simultáneamente online, a un desaprovechamiento del tiempo y esfuerzo que se requiere para resolver una tarea compartida.

Escenario: Imaginemos un grupo de desarrollo de software *open source* donde programadores, testers, diseñadores y el líder del proyecto se encuentran trabajando distribuidos geográficamente.

[1] Este patrón fue enviado y aceptado en el PloP 2004, realizado en Monticello, Illinois, USA. La metodología de trabajo para la elaboración del mismo fue de Shepherding. Los Shepherds fueron Doug Lee y Joe Yoder a quienes agradecemos sus aportes para mejorar este patrón.

El grupo está en constante comunicación vía e-mail. Dos horas antes de terminar el día, el equipo de programadores encuentra un problema que les impide continuar. Los programadores envían una descripción del problema vía e-mail al líder de diseño, único responsable de proveer la respuesta. Conociendo la naturaleza asincrónica del e-mail, posponen las actividades pendientes para el día siguiente. En el mismo momento que los programadores envían el e-mail, el líder de diseño se encuentra casualmente trabajando en su estación, sin embargo no presta atención al e-mail recibido, el cual se propone tratar mas tarde.
En este escenario, un problema que se podría haber solucionado de forma inmediata, se pospone hasta el próximo día y se pierden preciadas horas de trabajo.

Fuerzas: las fuerzas nos brindan los Beneficios, riesgos, perjuicios que puede ocasionar su uso.

- La colaboración en forma asincrónica es preciada porque ofrece a los participantes tiempo para elaborar sus contribuciones y la posibilidad de contribuir cuando les sea más oportuno.

- En ciertos casos la posibilidad de interactuar de forma sincrónica reduce el esfuerzo y el tiempo que se requiere para resolver una tarea compartida.

- La colaboración sincrónica requiere que los participantes se encuentren simultáneamente online. Ya sea casualmente o de forma intencional.

- La integración de servicios de colaboración, toma lo mejor de cada uno y de esta forma potencian la aplicación donde ellos estén integrados.

- Los servicios de colaboración sincrónica cuentan comúnmente con un indicador de presencia. En contraste, los servicios de groupware asincrónico carecen del mismo. El mecanismo de presencia provee información sobre la disponibilidad (libre, ocupado, desconectado, etc.) de los usuarios.

Solución: Extender el servicio de colaboración asincrónica con un mecanismo indicador de *presencia* en el/los servicio/s de colaboración sincrónica.

Participantes: Podemos identificar tres participantes

- Servicio de Groupware Asincrónico: Permite la colaboración de manera asincrónica. Ejemplos de servicios asincrónicos son: e-mail, foros, y web-logs. El groupware asincrónico posee comúnmente un mecanismo de identificación de usuarios. Por ejemplo: en el caso del e-mail se utiliza la dirección de e-mail.

- Servicio de Groupware Sincrónico: Permite la colaboración de manera sincrónica. Ejemplos de servicios sincrónicos son: el Chat, IM, y los whiteboards. Posee comúnmente un mecanismo de identificación de usuario. Por ejemplo: en el caso del Chat se utiliza el nickname.

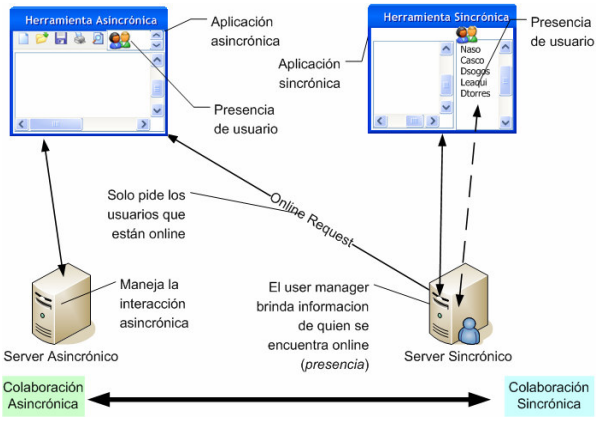


Ilustración 2

- Servicio de Presencia: Mantiene y provee información sobre el estado de los usuarios de un sistema de software. Ej.: Conectado, desconectado, ocupado, etc.

Colaboración entre servicios: El usuario se conecta y hace uso del servicio sincrónico, eventualmente cambia su estado y se desconecta. El servicio sincrónico comunica cambios de estado al servicio de presencia. A su vez, consulta el estado de otras personas para comunicarlo al usuario.

El usuario inicia el servicio asincrónico. Cada vez que el servicio asincrónico debe mostrar al usuario una contribución hecha por alguna otra persona, consulta el estado de la persona en los servicios de presencia disponibles, para mostrarlos junto con la contribución.
El usuario observa que el autor de una contribución se encuentra disponible (Por ejemplo: disponible para chatear) e inicia una interacción sincrónica

Detalles de Implementación: La arquitectura (Ilustración 2) se basa en la integración de servicios provistos por varios servidores, cada uno brindando servicios específicos. En nuestro caso un servidor de usuarios brinda el servicio de presencia. El estado de los usuarios, y el cambio del mismo, es comunicado por el servicio sincrónico al servicio de presencia. De este modo el servicio asincrónico es extendido usando el servicio de presencia para comunicar el estado de los usuarios y, de esta forma, potenciar la colaboración sincrónica. El servicio de presencia es el encargado de manipular la información del estado de los usuarios manteniendo un registro de los mismos. El mapeo entre el servicio asincrónico y el de presencia, para informar el estado del usuario, se realiza a través de la dirección de e-mail o en su defecto por algún nickname. Esto implica tener y respetar un protocolo[2] que sirva de interfaz entre los servicios.

Este patrón es mas fácil de implementar cuando: 1) los servicios a integrar son provistos por el mismo sistema. 2) los servicios a integrar son ofrecidos por sistemas independientes pero que comparten un mismo protocolo de comunicación y trabajando sobre la misma red.

Aplicación: Se debe usar este patrón cuando:

- Un grupo de usuarios que se encuentran distribuidos y utilizando groupware asincrónico, eventualmente deban realizar una tarea que requiera colaborar de manera sincrónica por un corto periodo de tiempo (Por ejemplo: ponerse de acuerdo sobre la interpretación de algún requerimiento). Y se desee mostrar la posibilidad de contactarse de manera sincrónica.

- Un grupo de usuarios estén trabajando sobre los mismos documentos (Los autores del mismo son más de uno). En algún momento se necesita contactar a alguno de los usuarios que trabajó sobre esos documentos para que clarifique algún punto que no queda claro. Se desea informar de la posibilidad de contactar al usuario para acelerar el proceso de clarificación.

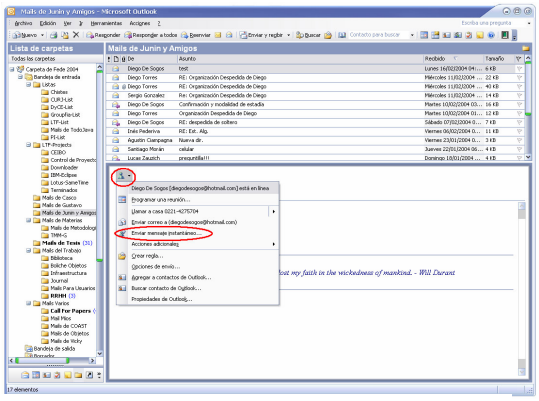


Ilustración 3

Consecuencias: Este patrón ofrece los siguientes beneficios:

- Reducir el tiempo y el esfuerzo usado en sincronizar usuarios que desean trabajar de manera sincrónica.

[2] Si bien actualmente cada servicio de mensajeria tiene su propio protocolo definido, se esta trabajando en la elaboración de protocolos estándares. Ej.: Jabber, IMMP.

- Ofrecer una visión de los servicios que se deben integrar para soportar la interacción entre las aplicaciones asincrónicas y las aplicaciones sincrónicas.

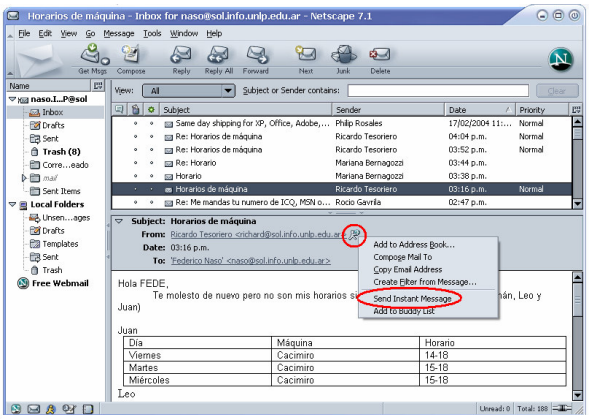


Ilustración 4

Ejemplos de uso: El cliente de mail de MS Outlook[3] integrado con ICQ[4]. El servicio de presencia es provisto por la red de ICQ. El mapeo entre los IDs usados por Outlook (e-mail address) y el ID usado por ICQ (en ICQ se usa UIN que significa *"Universal Internet Number"* o "número universal de Internet"). se lleva a cabo a través de la información de los contactos que posee ICQ. Un solo UIN de ICQ puede ser mapeado a muchas direcciones de mail. Cuando un mail es seleccionado para ser visto en el Outlook, se realiza una búsqueda a través del mapeo en la lista de contactos del ICQ. Si en la lista se encuentra el contacto se muestra el estado (Online, offline, etc) del contacto en el Outlook. De manera similar Outlook integra el cliente de mensajeria MSN Messenger (Ilustración 3). El servicio de presencia es provisto por la red de MSN network. El MSN Messenger[5] usa la dirección de e-mail de su propia red (ej: Hotmail.com msn.com y passport.com) como identificación de usuarios. Se puede asociar también una cuenta propia de e-mail a un passport (Por ejemplo: nombre@dominio.com).

Nestcape 7.1 se integra también con AOL Instant Messenger[6] (AIM) (Ilustración 4). Trabaja de manea similar a Microsoft Outlook con ICQ. Pero sin embargo, el mapeo (permite solo un e-mail por AIM screen name) es realizado a través de la libreta de direcciones de Netscape. Y con en el resto de los casos el servicio de presencia es provisto por la propia red AOL AIM Network.

Patrón - Notifier Broker

Intención: Notificar a usuarios de sistemas groupware, la disponibilidad de nuevo contenido o actualización del mismo.

Problema: La carencia de un mecanismo que informe, a usuarios o grupos de usuarios, sobre la disponibilidad de material o cambios en el mismo, conlleva a pérdida de tiempo y a uso ineficiente de la información disponible, ya que se debe informar por otros medios que no son parte del sistema.

Fuerzas:

- Los servicios de publicación no cuentan con un mecanismo automático que informe mediante eventos a los usuarios sobre cambios en el contenido o sobre nuevo contenido publicado.

- Los servicios de publicación son casi imprescindibles a la hora de compartir información y mantener un historial de la misma en

Ilustración 5

[3] http://www.microsoft.com/outlook/
[4] http://www.icq.com
[5] http://messenger.msn.com/
[6] http://www.aim.com

contextos de grupos distribuidos.

- Los servicios de comunicación brindan la posibilidad de mantener comunicados a personas geográficamente distantes.

- La comunicación, mediante eventos automáticos, entre grupos es crucial para mantener a los mismos informados de los cambios y decisiones establecidas en el momento que se realizan.

Solución: Integrar un servicio de comunicación que, mediante la programación de eventos automáticos, informen sobre la disponibilidad de nuevo, o de cambios, material publicado a usuarios registrados para que puedan hacer uso del mismo de manera inmediata.

Ejemplos de uso: Al ser un patrón de integración entre servicios diversos se pueden encontrar infinidad de ejemplos de uso. BSCW[7], IBM Lotus Team Workplace[8] (QuickPlace), Ms Sharepoint. Así por ejemplo podemos encontrar BSCW. BSCW es una de las implementaciones más completas (Ilustración 5), la misma permite compartir documentos, direcciones URL9, notas, etc. Los usuarios poseen un espacio de trabajo el cual puede ser compartido mediante una invitación a otros usuarios.

Cada vez que se realiza un cambio en algunos de los objetos publicados en BSCW, este informa mediante un mail a las personas interesadas en dicho objeto. Otros ejemplos como QuickPlace permiten la comunicación mediante Mensajes Instantaneos a los usuarios que esten online.

Patrón – Conference

Intención: Simular la interacción cara a cara entre personas o grupos de personas que se encuentran en diferentes lugares geográficamente distantes.

Problemas: Imposibilidad de llevar a cabo reuniones o conferencias entre personas o grupos de personas, debido a que las mismas se encuentran geográficamente distantes.

Fuerzas:

- Las conferencias remotas son preciadas porque no requieren que los participantes se encuentren en el mismo lugar físico.

- La posibilidad de interactuar en una video/audio conferencia remota reduce el tiempo y esfuerzo que se requiere para resolver un problema mediante otras vías, como por ejemplo el e-mail.

- Las conferencias remotas necesitan algún servicio de comunicación textual (chat) en el caso de que la comunicación por audio/video no sea fiable.

Ilustración 6

Solución: Utilizar la integración de los servicios de video, audio y de Chat como un paquete de comunicación integral para permitir la comunicación entre personas o grupos de personas que se encuentran geográficamente distantes.

---

[7] www.bscw.fit.org
[8] www.lotus.com
[9] URL: **U**niform **R**esource **L**ocator

Ejemplos de uso: Un ejemplo de este tipo de integración es EyeBall[10] (Ilustración 6) de Eyeball Networks Inc. EyeBall permite realizar simultáneamente conferencias utilizando audio, video o chat entre multiples usuarios. Eyeball esta orientado al Web, las aplicaciones están embebidas en la misma ventana. Además posee una pizarra compartida en donde los usuarios pueden realizar anotaciones con sus herramientas de dibujo.

Patrón - Gossip

Intención: Informar a usuarios que estén trabajando en un *espacio de trabajo* colaborativo sobre las actividades que se encuentran realizando otros usuarios dentro del mismo *espacio de trabajo*.

Problema: Usuarios que se encuentran trabajando en el mismo *espacio de trabajo* colaborativo, no se percatan de que hay otros trabajando al mismo tiempo y pierden la posibilidad de interactuar entre ellos.

Fuerzas:

- Los usuarios no se percatan de la existencia de otros usuarios en el mismo *espacio de trabajo*.
- Por no tener conocimiento de quien esta trabajando en el mismo *espacio de trabajo*, no existe una forma de sincronizar el trabajo entre ellos.
- Es deseable poseer cierta información de quien y que objetos se están manipulando en los distintos *espacios de trabajo*.

Solución: Utilizar algunos de los mecanismos de *presencia de usuario* para informar al resto sobre la disponibilidad de otros usuarios de trabajar en el mismo *espacio de trabajo*.

Ejemplos de uso: Dolphin[11] esta basado en un modelo de datos de hypermedia dentro de un espacio de trabajo público. Este modelo soporta la construcción de estructuras, como así también la representación de dependencias entre documentos.

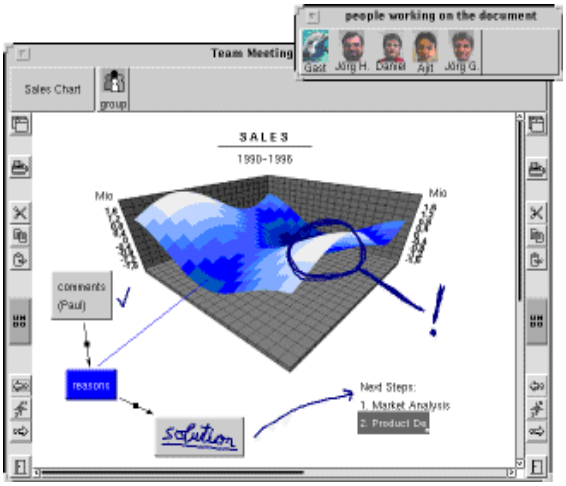


Ilustración 7

Otras implementaciones colaborativas en donde se use mecanismo de presencia son: SEPIA[12], ambiente que soporta producción de documentos de forma sincrónica y asincrónica entre autores que se encuentran en grupos distribuidos geográficamente; Vital[13] ("Virtual Teaching and Learning") es un ambiente hipermedial para la enseñanza y aprendizaje colaborativo. Vital utiliza el concepto de "rooms" en donde los participantes pueden hacer sus contribuciones. Y CHIPS[14] provee soporte hypermedial a procesos de desarrollo realizados en ambientes cooperativos.

[10] www.eyeball.com
[11] http://www.ipsi.fraunhofer.de/concert/activities/internal/dolphin.html
[12] http://www.ipsi.fraunhofer.de/concert/activities/past/sepia.html
[13] http://www.ipsi.fraunhofer.de/concert/activities/past/vital.html
[14] http://www.ipsi.fraunhofer.de/concert/activities/internal/chips.html

7. Conclusión

El uso de patrones es una práctica establecida para documentar en detalle problemas recurrentes y soluciones de validez probada. Los patrones incluidos aquí documentan problemas derivados de las necesidades de integración de servicios groupware identificadas recurrentemente en ambientes de groupware comerciales y de investigación. El detalle con el que se describe el problema de cada patrón es suficiente como para que el diseñador de groupware identifique el problema en los sistemas que diseña y pueda decidir sobre la aplicación de la solución propuesta por el patrón. Asimismo, los patrones aquí expuestos son una fuente de inspiración para quienes diseñan groupware.

En nuestro grupo, hemos comprobado que documentar patrones de integración reduce el esfuerzo de desarrollo de groupware, dado que nos ofrecen un punto de partida sobre el cual no es necesaria mucha discusión. Esto nos permite enfocar nuestras energías en buscar nuevas alternativas de integración de servicios para conseguir groupware innovador y más útil. Estas nuevas alternativas de integración de servicios, una vez probada su validez más allá de nuestros desarrollos, llegarán a estar documentadas en nuevos patrones.

Agradecimientos

En principio quisiéramos agradecer a los *Shepherds*, Doug Lee y Joe Yoder por sus invalorables aportes durante la etapa de *Shepherding* al pattern *Rendezvous* el cual fue presentado en el PLoP'04. También queremos agradecer a Diego Torres y Leandro Quiroga por sus aportes mediante comentarios realizados a este paper.

Referencias

[1]. http://www.usabilityfirst.com/groupware/

[2]. C. A. Ellis, Simon J. Gibbs, Gail L. Rein: Groupware: Some Issues and Experiences. Commun. ACM 34(1): 39-58 (1991)

[3]. Robert Slagter, Dynamic Groupware Services. Modular design of tailorable groupware.. ISBN: 90-75176-37-6. Telematica Institut, The Netherlands. 2004

[4]. Vissers, C. A., Ferreira Pires, L., Quartel, D. A. C., & Van Sinderen, M. J. (2002). The architectural design of distributed systems: Reader for the design of Telematics Systems. Enschede, The Netherlands: University of Twente.

[5]. Roseman, M. and Greenberg, S. . *Simplifying Component Development in an Integrated Groupware Environment.* Proceedings of UIST '97.

[6]. Carl Gutwin and Saul Greenberg, Workspace Awareness for Groupware, CHI '96 Companion, Vancouver, BC Canada, 1996 ACM 0-89791-832.

[7]. C. A. Ellis, S. J. Gibbs, Concurrency Control in Groupware Systems. ACM 0-89791-317-5/89/0005/0399. 1989

[8] . http://bscw.fit.fraunhofer.de/

[9]. http://www.markroseman.com/teamwave/

[10]. http://www.groupkit.org/

[11]. Tietze, Daniel A. A Framework for Developing Component-based Co-Operative Applications., GMD Research Series Nr.7/2001, 2001, ISBN: 3-88457-390-X, Dissertation at Darmstadt University of Technology

[12]. Roseman, M and Greenberg, S. TeamRooms: Network Places for Colaboration. Proc. of ACM CSCW ´96. October, 1996.

[13]. Yann Laurillan et al. *Clover Architecture for Groupware* CSCW 2002 November 16-20, 2002. New Orleáns, Luisiana, USA. Proc. of ACM CSCW '02

[14]. ftp://ftp.rfc-editor.org/in-notes/rfc822.txt

[15]. C. Alexander, The Timeless Way of Building, Oxford University Press, New York, 1979

[16]. C. Alexander, el al: "A Pattern Language", Oxford University Press, 1977

[17]. Peter Coad, "Object-Oriented Patterns". Communications of the ACM. 35(9):152-159, September 1992

[18]. Peter Coad, D. North and M. Mayfield. Object Models: strategies, patterns and applications, Prentice Hall, Englewood Cliffs, 1995

[19]. Fowler, M. Analysis Patterns: Reusable Object Models. Addison-Wesley, Reading, Mass. 1997.

[20]. http://www.comp-lancs.ac.uk/computing/research/cseg/projects/pointer/pointer.html

[21]. http://swiki.darmstadt.gmd.de/gw-patterns/

[22]. http://hci.ethz.ch/patterns

[23]. http://timetripper.com/uipatterns

[24]. http://hillside.net/conferences/plop.htm

[25]. Gamma E. et al. *Design Patterns: Elements of reusable Object-Oriented Software*. Reading, Mass, Addison-Weskey. 1995

[26]. http://iserverd1.khstu.ru/oscar/

[27]. http://www.jabber.org/protocol/

[28]. http://messenger.yahoo.com

[29].http://www.ietf.org/html.charters/OLD/impp-charter.html

Non-deterministic Regular Positive Negative Inference *NRPNI* *

Gloria Inés Alvarez
Pontificia Universidad Javeriana - Seccional Cali, Grupo DESTINO,
Cali, Colombia
galvarez@dsic.upv.es

and

José Ruiz
Universidad Politécnica de Valencia, DSIC,
Valencia, Spain, 46022
jruiz@dsic.upv.es

and

Antonio Cano
Universidad Politécnica de Valencia, DSIC,
Valencia, Spain, 46022
acano@dsic.upv.es

and

Pedro García
Universidad Politécnica de Valencia, DSIC,
Valencia, Spain, 46022
pgarcia@dsic.upv.es

Abstract

We propose an algorithm, called *NRPNI*, for inference of regular languages. On input of a sample from the target language, it outputs a nondeterministic finite automata consistent with the input sample. This algorithm converges to an automaton of size not greater than the size of the minimal deterministic automaton of the target language in the limit. The algorithm makes use of the concept of residual finite automaton and it has polynomial temporal complexity.

Keywords: Grammatical inference, Residual Finite State Automata (RFSA)

1 Introduction

Most of the algorithms for regular grammatical inference reported so far output Deterministic Finite Automata. One of the most widely known is the *Regular Positive and Negative Inference algorithm (RPNI)* [9] which, on input of a finite sample of the target language, it outputs a deterministic finite automaton, which is consistent with the sample and converges to the minimal automaton recognizing the language in polynomial time.

*Work partially supported by Spanish CICYT under TIC2003-09319-C03-02

Denis et al. [1, 2, 3] define the concept of *Residual Finite State Automata* (RFSA) as a nondeterministic automata such that every state is associated to a residual language of the language it recognizes. They propose the algorithm *DeLeTe2* such that its input is a sample of the target language L, and its output converges to an RFSA whose size is between the size of the canonical RFSA and the size of the minimal DFA that accepts L. An argument in favor of the use of an RFSA representation for target languages is that sometimes the size of an RFSA is smaller than the minimal DFA for the same language. The only difficulty is that the output of the algorithm may not be consistent with the input sample if the sample is not characteristic. If one studies this algorithm, one can observe that it can be explained as a modification of *Gold*'s Algorithm [6, 10].

Considering the relations existing between *Gold*'s and *RPNI* algorithms [4], we propose the algorithm *NRPNI*, a nondeterministic version of *RPNI* algorithm which inherits from the classical one some important features, as consistency with the input, convergence in the limit and polynomial complexity. It also incorporates the concept of residual automata and its properties, and thus outputs nondeterministic automata. For a given sample, if *NRPNI* algorithm does not find a consistent hypothesis of smaller size than *RPNI* would do, it outputs the same hypothesis than *RPNI*.

The rest of the paper is structured as follows: section 2 contains some definitions that will be used in the rest of the paper. In section 3, we give a description of *Gold*'s and *DeLeTe2* algorithms, as well as two variations of the former algorithm, one of them being equivalent to *DeLeTe2*. In section 4 we give a description of *RPNI* algorithm. Section 5 contains *NRPNI*, a nondeterministic extension of *RPNI*. For better understanding, an example of the algorithms is shown in Section 6. The article ends with the conclusions.

2 Definitions and notation

Definitions not contained in this section can be found in [7, 10]. Definitions and previous works concerning RFSAs can be found in [1, 2, 3].

2.1 Formal languages

Let A be a finite alphabet and let A^* be the free monoid generated by A with concatenation as the internal operation and ε as neutral element. A *language* L over A is a subset of A^*. The elements of L are called *words*. The length of a word $w \in A^*$ is noted $|w|$. Given $x \in A^*$, if $x = uv$ with $u, v \in A^*$, then u (resp. v) is called *prefix* (resp. *suffix*) of x. $\Pr(L)$ (resp. $\mathrm{Suf}(L)$) denotes the set of prefixes (resp. suffixes) of L. The product of two languages $L_1, L_2 \subseteq A^*$ is defined as: $L_1 \cdot L_2 = \{u_1u_2 | u_1 \in L_1 \wedge u_2 \in L_2\}$. Sometimes $L_1 \cdot L_2$ will be denoted simply as L_1L_2. Throughout the paper, the *lexicographical order* in A^* will be denoted as $\ll$. Assuming that A is totally ordered by $<$ and given $u, v \in A^*$ with $u = u_1 \dots u_m$ and $v = v_1 \dots v_n$, $u \ll v$ if and only if ($|u| < |v|$) or ($|u| = |v|$ and $\exists j$, $1 \leq j \leq n, m$ such that $u_1 \dots u_j = v_1 \dots v_j$ and $u_{j+1} < v_{j+1}$).

A *Nondeterministic Finite Automaton* (NFA) is a 5-tuple $\mathcal{A} = (Q, A, \delta, I, F)$ where Q is the (finite) set of states, A is a finite alphabet, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states and δ is a partial function that maps $Q \times A$ in 2^Q. The extension of this function to words is also denoted δ. A word x is accepted by $\mathcal{A}$ if $\delta(I, x) \cap F \neq \emptyset$. The set of words accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A})$. A *Deterministic Finite Automaton* (DFA) is a NFA such that $|I| = 1$ and $|\delta(q, a)| \leq 1$, $\forall q \in Q$, $\forall a \in A$. Two automata are *equivalent* if they recognize the same language.

Given an automaton $\mathcal{A} = (Q, A, \delta, I, F)$ and a set $P \subseteq Q$, the *restriction* of $\mathcal{A}$ to P is the automaton $\mathcal{A}_P = (P, A, \delta', I', F')$, where $I' = I \cap P$, $F' = F \cap P$ and δ' is the restriction of δ that maps $P \times A$ in 2^P.

Given a finite set of words D_+, the *prefix tree acceptor of* D_+ is defined as the automaton $\mathcal{A} = (Q, A, \delta, q_0, F)$ where $Q = Pr(D_+)$, $q_0 = \varepsilon$, $F = D_+$ and $\delta(u, a) = ua$, $\forall u, ua \in Q$.

A Moore machine is a 6-tuple $M = (Q, A, B, \delta, q_0, \Phi)$, where A (resp. B) is the input (resp. output) alphabet, δ is a partial function that maps $Q \times A$ in Q and Φ is a function that maps Q in B called *output function*. Throughout this paper $B = \{0, 1, ?\}$. A nondeterministic Moore machine is defined in a similar way except for the fact that δ maps $Q \times A$ in 2^Q and the set of initial states is I. The automaton related to a Moore machine $M = (Q, A, B, \delta, I, \Phi)$ is $\mathcal{A} = (Q, A, \delta, I, F)$ where $F = \{q \in Q : \Phi(q) = 1\}$. The *restriction* of M to $P \subseteq Q$ is the machine M_P defined as in the case of automata.

The behavior of M is given by the partial function $t_M : A^* \to B$ defined as $t_M(x) = \Phi(\delta(q_0, x))$, for every $x \in A^*$ such that $\delta(q_0, x)$ is defined.

Given two disjoint finite sets of words D_+ and D_-, we define the (D_+, D_-)-*Prefix Tree Moore Machine* ($PTMM(D_+, D_-)$) as the Moore machine having $B = \{0, 1, ?\}$, $Q = \Pr(D_+ \cup D_-)$, $q_0 = \varepsilon$ and $\delta(u, a) = ua$

if $u, ua \in Q$ and $a \in A$. For every state u, the value of the output function associated to u is 1, 0 or ? (undefined) depending whether u belongs to D_+, to D_- or to $Q - (D_+ \cup D_-)$ respectively. The size of the sample (D_+, D_-) is $\sum_{w \in D_+ \cup D_-} |w|$.

A Moore machine $M = (Q, A, \{0, 1, ?\}, \delta, q_0, \Phi)$ is *consistent* with (D_+, D_-) if $\forall x \in D_+$ we have $\Phi(x) = 1$ and $\forall x \in D_-$ we have $\Phi(x) = 0$.

2.2 Residual Finite State Automata (RFSA)

The *derivative* of a language L by a word u, also called *residual language* of L associated to u is $u^{-1}L = \{v \in A^* : uv \in L\}$. A residual language $u^{-1}L$ is *composite* if $u^{-1}L = \cup_{v^{-1}L \subsetneq u^{-1}L} v^{-1}L$. A residual language is *prime* if it is not composite.

If $\mathcal{A} = (Q, A, \delta, I, F)$ is an NFA and $q \in Q$, we define the language accepted in automaton $\mathcal{A}$ from state q as $L(\mathcal{A}, q) = \{v \in A^* : \delta(q, v) \cap F \neq \emptyset\}$.

A Residual Finite State Automata RFSA [2] is an automaton $\mathcal{A} = \langle Q, A, \delta, I, F \rangle$ such that, for each $q \in Q$, $L(\mathcal{A}, q)$ is a residual language of the language L recognized by $\mathcal{A}$. So $\forall q \in Q, \exists u \in A^*$ such that $L(\mathcal{A}, q) = u^{-1}L$. In other words, a *Residual Finite State Automaton* (RFSA) $\mathcal{A}$ is an NFA such that every state defines a residual language of $L(\mathcal{A})$.

Two operators on RFSAs, that preserve equivalency, are defined [2]. The saturation and reduction operators. Given $\mathcal{A} = (Q, A, \delta, I, F)$ the *saturated automaton* of $\mathcal{A}$ is the automaton $\mathcal{A}^s = (Q, A, \delta^s, I^s, F)$, where $I^s = \{q \in Q : L(\mathcal{A}, q) \subseteq L(\mathcal{A})\}$ and $\forall q \in Q$, $\forall a \in A$, $\delta^s(q, a) = \{q' \in Q : L(\mathcal{A}, q') \subseteq a^{-1}L(\mathcal{A}, q)\}$. If in automaton $\mathcal{A}$ all the residual languages (not only the prime ones) are considered as states, the new automata is known as saturated RFSA of the minimal DFA for L. The *reduction* operator allows to eliminate from an automaton $\mathcal{A}^s$ the composite states and the transitions related to them. Both operations are useful to get the *canonical RFSA* associated with $\mathcal{A}$: first the saturation operator is applied to $\mathcal{A}$, and then the reduction operator is applied to the result. It is known [2] that every regular language L is recognized by a unique reduced saturated RFSA, the *canonical RFSA* of L.

Formally, given a language $L \subseteq A^*$ the *canonical* RFSA of L is the automaton $\mathcal{A} = (Q, A, \delta, I, F)$ where:

- $Q = \{u^{-1}L : u^{-1}L \text{ is prime}, u \in A^*\}$
- A is the alphabet of L
- $\delta(u^{-1}L, a) = \{v^{-1}L \in Q : v^{-1}L \subseteq (ua)^{-1}L\}$
- $I = \{u^{-1}L \in Q : u^{-1}L \subseteq L\}$
- $F = \{u^{-1}L \in Q : \varepsilon \in u^{-1}L\}$

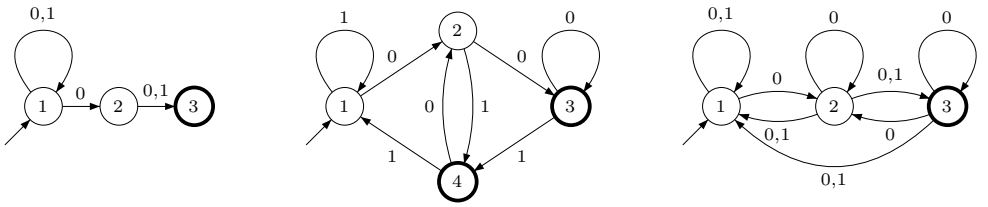
Two relations defined in the set of states of an automaton link RFSAs with grammatical inference. Let $D = (D_+, D_-)$ be a sample, let $u, v \in Pr(D_+)$. We say that $u \prec v$ if no word w exists such that $uw \in D_+$ and $vw \in D_-$. We say that $u \simeq v$ [1] if $u \prec v$ and $v \prec u$.

2.2.1 Example of RFSAs

The following example has been taken from [2]. Let $A = \{0, 1\}$ and let $L = A^*0A$. L can be recognized by the three automata of Figure 1. States that output 1 (resp. 0) are drawn using thick (resp. thin) lines.

- The first one $\mathcal{A}_1$ is neither DFA nor RFSA. The languages associated with states are: $L(\mathcal{A}_1, 1) = A^*0A$, $L(\mathcal{A}_1, 2) = A$ and $L(\mathcal{A}_1, 3) = \varepsilon$. One can see that $L(\mathcal{A}_1, 3) = \varepsilon$ and $\not\exists u \in A^*$ such that $L(\mathcal{A}_1, 3) = u^{-1}L$.
- Automaton $\mathcal{A}_2$ is a minimal automaton recognizing L and thus, is an RFSA; in this case $L(\mathcal{A}_2, 1) = A^*0A$, $L(\mathcal{A}_2, 2) = A^*0A + A$, $L(\mathcal{A}_2, 3) = A^*0A + A + \varepsilon$, $L(\mathcal{A}_2, 4) = A^*0A + \varepsilon$.
- Automaton $\mathcal{A}_3$ is the L's canonical RFSA, which is not a DFA. The languages associated with states are: $L(\mathcal{A}_3, 1) = \varepsilon^{-1}L$, $L(\mathcal{A}_3, 2) = 0^{-1}L$ and $L(\mathcal{A}_3, 3) = 01^{-1}L$.

[1] This relation is known in the terminology set up by Gold as *not obviously different states*. Other authors call it *compatible states*



$\mathcal{A}_1$ $\mathcal{A}_2$ $\mathcal{A}_3$

Figure 1: $\mathcal{A}_1$ is an automaton recognizing $L = A^*0A$ which is neither a NFA nor a RFSA. $\mathcal{A}_2$ is a DFA recognizing L which is also RFSA. $\mathcal{A}_3$ is the canonical RFSA for L.

2.3 Grammatical Inference

The aim of grammatical inference is to obtain a description of a language L by means of a sample (a set of words labelled as belonging to L or to its complement). Throughout this work we will suppose that the target language L is regular so the model we will look for is an automaton.

Among the concepts required in grammatical inference is concept of convergence. One of the convergence criteria is *identification in the limit*, introduced by Gold [5]. An algorithm A reads words from an input sample $D = (D_+, D_-)$, where the words of D_+ belong to the target language and the words of D_- to its complement, and outputs a representative (NFA in this case) as hypothesis. A identifies the regular language L in the limit if for any presentation of L, the sequence of hypothesis converges to an automaton that accepts L.

3 *Gold*'s and *DeLeTe2* algorithms

The classical *Gold* algorithm converges to the minimal deterministic automaton that recognizes the target language L in the limit [5]. On the other hand, *DeLeTe2* algorithm [3] is used to infer regular languages described by nondeterministic finite automata. This section briefly describes both algorithms and the relationship between them.

The algorithm proposed by Gold [6][10] is shown in algorithm 1; it has three parts: the first one (lines 3-6) chooses the states S of the hypothesis, that is, the subset of states of M that are distinguishable from all the others in $Pr(D_+ \cup D_-)$. The second part builds a new Moore machine M' with the states in S, and the transitions calculated in lines 10-16; the new transitions reflect in the hypothesis the effect of the not obviously different states, which are grouped and represented by one of them in S. The third part returns the answer according with the consistency of the new machine M'. If M' is consistent with the input sample, it is returned as answer, else the initial Prefix Tree Moore Machine is returned. A deeper description of this algorithm can be found in [4].

DeLeTe2 outputs residual nondeterministic finite automata (RFSA), based on the concept of residual languages. Every state of a RFSA is associated to a residual language of the language it recognizes. As have been said, a residual language is composite if it is the union of residuals of the same language contained in it, otherwise it is called prime. The algorithm looks for inclusion relations between the residual languages and reflects those situations using a saturation operator. As it can be expected, the method becomes more interesting when the target automaton contains many composite residual languages, because then it may exist many inclusion relations between states that make the size of the hypothesis to decrease. Otherwise, if most of the residual languages of the target automaton are prime the output automaton would have a size similar to the size of the minimal DFA [3]. The *DeLeTe2* algorithm usually infers automata which are not consistent with the sample, although in those cases changes can be made that make the algorithm to output a consistent automaton without affecting the correct convergence.

3.1 *NGold1* algorithm

We propose the following modifications to *Gold* (D_+, D_-) algorithm, in order to obtain an algorithm that although it inherits from the original one the fact that it does not always obtain consistent automata, it will converge to the saturated RFSA of the minimum DFA that recognizes the target language:

Algorithm 1 $Gold(D_+, D_-)$

1: $M := PTMM(D_+, D_-)$ // $M = (Pr(D_+, D_-), A, \delta_0, q_0, \Phi)$
2: $S := \{\varepsilon\}$
3: **while** $\exists s' \in SA - S$ such that $\forall s \in S$, $s \not\simeq s'$ **do**
4: Choose s' // The first one in lexicographical order
5: $S := S \cup \{s'\}$
6: **end while**
7: $Q := S$
8: $q_0 := \{\varepsilon\}$
9: **for** $s \in S$ **do**
10: **for** $a \in A$ **do**
11: **if** $sa \in S$ **then**
12: $\delta(s, a) := sa$
13: **else**
14: $\delta(s, a) :=$ first $s' \in S$ such that $sa \simeq s'$ // In lexicographical order
15: **end if**
16: **end for**
17: **end for**
18: $M' := (Q, A, \delta, \varepsilon, \Phi)$
19: **if** M' is consistent with (D_+, D_-) **then**
20: Return M'
21: **else**
22: Return M
23: **end if**

- The line 8 in the original algorithm (algorithm 1) has to be changed to $I = \{u \in S : u \prec \varepsilon\}$.
- The loop beginning in line 10 has to be changed to:

for $a \in A$ **do**
$\delta(s, a) = \{s' \in S : s' \prec sa\}$
end for

We will refer to this algorithm as *NGold1*. Note that the proposed changes do not substantially affect the time complexity with respect to the original algorithm, as checking for the relation $\prec$ is equivalent to checking for $\simeq$ and thus, the only difference is the greater amount of transitions that the RFSA may have with respect to a DFA with the same number of states.

Proposition 1 *Algorithm NGold1 converges to the saturated RFSA of the minimum DFA that recognizes the target language.*

Proof. *NGold1* works in the same way as the classical $Gold(D_+, D_-)$ algorithm while building the set S, so it distinguishes the states of the target automaton. On the other hand, if $u_1 \prec u_2$ and $u_1 \not\simeq u_2$, a word v will exist such that $u_1 v \in D_- \wedge u_2 v \in D_+$ so the relations "$\prec$" between states can also be established ■

3.2 *NGold2* algorithm

The process of building the states of the output automaton in *NGold1* algorithm continues until there is no state in $SA - S$ obviously different from every state of S. The hypothesis is then emitted. The following changes in *NGold1* give an algorithm that behaves exactly as the algorithm *DeLeTe2* proposed by Denis et al. [3]:

Every time we obtain a new version of the S set in *NGold1* algorithm we:

- Emit a new hypothesis

- Check it for consistency. If the hypothesis is not consistent and $SA - S$ is not empty we add a state to S and we build the following hypothesis. If it is consistent or $SA - S$ is empty, we finish.

We will refer to this algorithm as *NGold2*.

Proposition 2 *NGold2 and DeLeTe2 [3] are equivalent*

Proof. The relation $\simeq$ defined in [3] between states is equivalent to the relation known in Gold's terminology as *not obviously different* states. The process followed in both algorithms to add new states to the automaton is also equivalent. ■

4 The *RPNI* algorithm

RPNI (Regular Positive and Negative Inference) algorithm [8, 9] receives a sample of the target sample as input and it outputs, in polynomial time, a DFA consistent with the input.

Algorithm *RPNI* (D_+, D_-) (see algorithm 2) begins building the Prefix Tree Moore Machine of the sample, then it merges every state with the previous ones in lexicographical order and propagates the merges to keep a deterministic automaton under the condition that it does not accept a negative sample. The merging of states is done by the function *detmerge* (see algorithm 3) which guarantees the consistency of the intermediate automata and thus, of the output automaton. During this process, the function *merge* is called to merge states and rearrange the transitions in the automata. Function *detmerge*, also redefines the output function of the merged states if the output of one of them has not been previously defined. That is the main difference between the behavior of *Gold*'s and *RPNI* algorithms; while the former never changes the output function of a state in the Moore machine, the later may change it from undefined to either positive or negative.

Algorithm 2 $RPNI(D_+, D_-)$

```
1:  M := PTMM(D+, D-)
2:  list := {u0, u1, ..., ur} //states of M in lexicographical order, u0 = λ//
3:  list' := {u1, ..., ur}
4:  q := u1
5:  while list' ≠ ∅ do
6:    for p ∈ list and p ≪ q (in order) do
7:      if detmerge(M, p, q) ≠ M then
8:        exit for
9:      end if
10:     M := detmerge(M, p, q)
11:   end for
12:   list := Delete from list the states which are not in M
13:   list' := Delete from list' the states which are not in M
14:   q := first(list')
15: end while
16: Return M
```

4.1 Temporal complexity of the *RPNI* algorithm

The complexity of the *RPNI* algorithm can be expressed in terms of the size of the sample n; by analyzing algorithm 2, it can be seen that *RPNI* is $O(n^3)$ because, in the worst case, the prefix acceptor tree will have as many nodes as there are symbols in the sample. Therefore the cycles in lines 5 and 6 of algorithm 2 have, in the worst case, a cost $O(n^2)$ and the call to *detmerge* has a cost $O(n)$ as can be seen in algorithm 3.

5 The *NRPNI* algorithm

NRPNI (Non-deterministic Regular Positive Negative Inference) algorithm uses *RPNI* strategy of merging states but it can infer nondeterministic automata while it keeps the property of being consistent with the

Algorithm 3 $detmerge(M, p, q)$ // $p \ll q$ in lexicographical order

```
1: M' := M
2: list := {(p, q)}
3: while list ≠ ∅ do
4:    (r, s) := First(list)
5:    M1 := merge(M', r, s)
6:    if M1 = M' then
7:       Return M
8:    else
9:       M' := M1
10:      for a ∈ A do
11:         if δ(p, a) and δ(q, a) are defined then
12:            list := apppend(list, (δ(p, a), δ(q, a)))
13:         end if
14:      end for
15:   end if
16: end while
17: Return M'
```

input sample. For a given sample, if *NRPNI* algorithm does not find a consistent hypothesis of smaller size than *RPNI* would do, it outputs the same hypothesis that *RPNI*.

5.1 Description of *NRPNI* algorithm

NRPNI algorithm (algorithm 4) works from a set of positive and negative samples (D_+, D_-) over an alphabet A of the target language. It initially applies the *RPNI* algorithm to the sample, the hypothesis obtained is called M. Then, it tries to find a smaller hypothesis, consistent with the sample, by including new transitions in the hypothesis machine; the new edges may become a deterministic machine in a non-deterministic one.

A copy of the *RPNI*'s answer is saved in M_1, whereas M becomes a tree that keeps just the transitions in the minimal length path to reach each state from the initial state q_0 (Figure 6 shows the Minimal Length Path, MLP, corresponding to *RPNI*'s answer of the example in the next section). Machine M will be used to calculate S and $SA - S$ sets, whereas M_1 will be the base for potentially smaller hypothesis. The set of states of the current hypothesis will be denoted as S (hence, at the beginning $S = \{q_0\}$). The set of transitions of the current hypothesis is formed by the transitions of M and it is extended with new edges generated because of precedence relationship between elements p and q (line 6). If there exists $p \in S$ and $q \in SA - S$ such that $q \prec p$, for each transition that ends in the state q a new transition must be added with the same origin but ending in p. In lines 8-9, the algorithm supposes the existence of the relation $q \prec p$ and it adds the new edges; if the automaton is still consistent with the negative samples D_- it is understood that the relation $q \prec p$ holds, and it proceeds to find out if state q is an initial state (lines 10-12). In any other case, the new edges are discarded and the opposite relationship is tried ($p \prec q$) as can be seen in lines 14-20. These comparisons are done for each element of the frontier $SA - S$ which is compared with each element of the current hypothesis S. Once all the frontier $SA - S$ is compared, a new hypothesis M'' is obtained. If M'' is consistent with the sample D_+, D_- this is the answer; in any other case, a new state is added to the hypothesis S: the least element of the frontier $SA - S$, in lexicographical order, is added and a new iteration begins. If the size of the hypothesis S reaches the size of the original hypothesis produced by *RPNI* algorithm, it outputs the same automata as *RPNI* would have returned.

5.2 Convergence of *NRPNI* algorithm

The convergence of *NRPNI* algorithm relies in the assertion that any characteristic sample for *RPNI* makes *NRPNI* to converge also. In fact, given the minimal DFA for a target language, if we call S_P (short prefixes) to the set of the smallest words in lexicographical order that reaches every state from the initial one, and K (kernel) to the set $S_P \cup S_P A$, where A is the alphabet, the characteristic sample must contain the words that:

Algorithm 4 $NRPNI(D_+, D_-)$

1: $M := RPNI(D_+, D_-)$ // $M = (Q, A, q_0, F, \delta)$
2: $M_1 := M$ // $M_1 = (Q_1, A, I_1, F_1, \delta_1)$
3: $M := MLP(M)$ // Minimum Length Paths from q_0 to each state
4: $S := \{q_0\}$; $I = \{q_0\}$; $\delta' := \delta_1$
5: **while** $|S| < |Q_1|$ **do**
6: **for** $(p, q) : p \in S, q \in SA - S$ // In lexicographical order on S,$SA - S$ and $p \ll q$ **do**
7: // It tries $q \prec p$ relationship
8: $\delta' = \delta' \cup \{(r, a, q) \in \delta' : (r, a, p) \in \delta', r \in Q_1, a \in A\}$
9: **if** $M' = (Q_1, A, I, F_1, \delta')$ is consistent with D_- **then**
10: **if** $p = \varepsilon$ **then**
11: $I = I \cup \{q\}$
12: **end if**
13: **else**
14: $\delta' := \delta' - \{(r, a, q) \in \delta' : (r, a, p) \in \delta', r \in Q_1, a \in A\}$
15: // It tries $p \prec q$ relationship
16: $\delta' = \delta' \cup \{(r, a, p) \in \delta' : (r, a, q) \in \delta', r \in Q_1, a \in A\}$
17: **if** $M' = (Q_1, A, I, F_1, \delta')$ is not consistent with D_- **then**
18: $\delta' := \delta' - \{(r, a, p) \in \delta' : (r, a, q) \in \delta', r \in Q_1, a \in A\}$
19: **end if**
20: **end if**
21: **end for**
22: M" =Subautomaton induced by S in $(Q_1, A, I, F_1, \delta')$
23: **if** M" is consistent with D_+, D_- **then**
24: Return M"
25: **else**
26: $S := S \cup first(SA - S \cap Q)$ // In lexicographical order
27: **end if**
28: **end while**
29: Return M_1

- For every element of K there must be a completion in the target language (in the case that the word reaches a final state, this completion must be ε).
- For every pair $(u, v) \in S_P \times K$ there must be a word w such that $(uw, vw) \in (D_+ \times D_-) \cup (D_- \times D_+)$.

To distinguish states u, v such that $u \prec v$ ($L_u \subseteq L_v$) we must have a word w such that $vw \in D_+$ and $uw \in D_-$ and we must never have w such that $uw \in D_+$ and $vw \in D_-$. So if two states are distinguished with respect to equivalence, they are also distinguished with respect to the precedence relation.

The worst case is the one in which there are not composite states. In this case it will be necessary to distinguish all the states. The convergence of *RPNI* guarantees this fact.

Algorithm *NRPNI* converges to a RFSA of a size smaller than or equal to the minimal DFA of the target language and greater than or equal to the size of the canonical RFSA. The first statement comes from the fact that *NRPNI* will never output an automaton of size greater than *RPNI* would do. The second arises from the fact that there may be composite states which, in lexicographical order, are previous to the last prime state in the output automaton.

5.3 Temporal Complexity of the *NRPNI* algorithm

NRPNI's temporal complexity is related to *RPNI*'s. In fact, we will see that both algorithms have the same upper bound: $O(n^3)$.

The first step *NRPNI* does is to call to *RPNI*. The extra processing done by *NRPNI* (see algorithm 4), has a worst case cost of $O(n^2)$ because of the $O(n)$ cost of the while loop in line 5 and the $O(n)$ cost of the consistency checking in line 17. So, the complete *NRPNI* algorithm has a worst case temporal complexity of $O(n^3)$. The worst case takes place when there are no composite states in the initial Moore machine and all the iterations are done without getting any consistent hypothesis. Calculating a more accurate upper bound for the temporal complexity of algorithm *NRPNI*, as function of the size of the automaton, is difficult because the number of transitions may increase with respect to the automaton output by *RPNI*, but the number of states may decrease.

6 Example

We are going to describe the behavior of algorithms *DeLeTe2*, *RPNI* and *NRPNI* by using a small sample that gives different outputs for the three algorithms.

Let $D_+ = \{\varepsilon, 00, 10, 11, 010\}$ and $D_- = \{0, 1, 001\}$; the corresponding Prefix Tree Moore Machine is depicted in Figure 2, where states that output 1 (resp. 0) are drawn using thick (resp. thin) lines whereas indeterminate states are drawn using dashed lines.

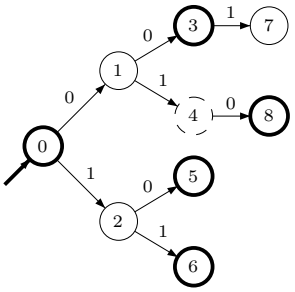


Figure 2: Prefix Tree Moore Machine

We find the following relations between states: 0 and 1 on one hand and 0 and 3 on the other are distinct, whereas $1 \simeq 2$, $0 \simeq 3$ and $1 \simeq 4$. We also find $1 \prec 3$ and $0 \prec 4$. Algorithm *DeLeTe* outputs automaton of Figure 3, which is not consistent with the sample (it accepts 001).

DeLeTe2 algorithm gives an output which is consistent with the input. It is depicted in Figure 4.

With the same input as before, *RPNI* algorithm merges the states 1 and 2, 0 and 3 and finally 0 and 8. It outputs the deterministic automaton of Figure 5 which, as expected, is consistent with the sample.

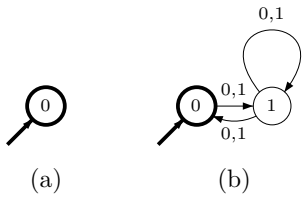


Figure 3: Successive hypothesis output by *DeLeTe* algorithm.

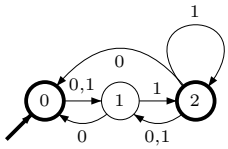


Figure 4: Output automata given by *DeLeTe2* algorithm.

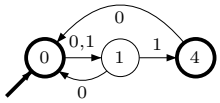


Figure 5: Output automata given by *RPNI* algorithm.

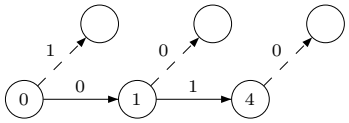


Figure 6: Minimal Length Path $MLP(M)$ related with automata in figure 2.

Finally *NRPNI* starts emitting the hypothesis of Figure 7 (a) which is not consistent. Then $S = \{0, 1\}$. It then merges the states 1 and 2 and finds the following relations in the set $SA \cup S$: $0 \simeq 3, 1 \prec 3, 0 \prec 4$ and $1 \prec 4$. It makes the hypothesis of 7 (b) which is consistent, so the algorithm ends. You should observe that the size of the output is smaller than the output given by *RPNI* and by *DeLeTe2* algorithms.

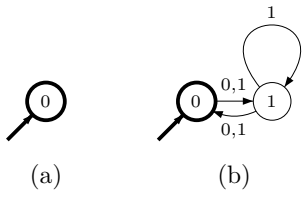


Figure 7: Successive hypothesis given by *NRPNI* algorithm.

Conclusions

Two important algorithms for automata inference are *RPNI* [9] and *Gold*'s algorithms [5, 6]. They both output DFAs, and the main difference between them is that the former always outputs consistent hypothesis, whereas the latter may not.

Recently Denis et al. [1, 2, 3] have used the concept of RFSAs to infer regular languages represented by NFAs which may be smaller in size to their equivalent minimal DFA. In this paper we showed that their *DeLeTe2* algorithm can be seen as an extension of *Gold*'s algorithm. We also proposed algorithm *NRPNI* which may output NFAs of a size smaller than the deterministic automata returned by *RPNI*. The proposed algorithm inherits from the *RPNI* polynomial complexity, consistency with the training sample and convergence in the limit to an automaton of size not greater than the size of the minimal deterministic automata for the target language.

References

[1] Denis, F. Lemay, A. and Terlutte, A. *Learning regular languages using nondeterministic finite automata.*A.L. Oliveira (Ed.), ICGI 2000, LNAI 1891, pp 39-50 (2000).

[2] Denis, F. Lemay, A. and Terlutte, A. *Residual finite state automata.*In STACS 2001, LNAI 2010, pp 144-157 (2001).

[3] Denis, F. Lemay, A. and Terlutte, A. *Learning regular languages using RFSAs.* Theoretical Computer Science 313(2), pp 267-294 (2004).

[4] García, P. Cano, A. and Ruiz, J. *A comparative study of two algorithms for Automata Identification.* A.L. Oliveira (Ed.), ICGI 2000, LNAI 1891, pp 115-126 (2000).

[5] Gold, E.M. *Language identification in the limit.* Information and Control 10, pp 447-474 (1967).

[6] Gold, E.M. *Complexity of Automaton Identification from Given Data.* Information and Control 37, pp 302-320 (1978).

[7] Hopcroft, J. and Ullman, J. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley (1979).

[8] Lang , K.J. *Random DFAs can be Approximately Learned from Sparse Uniform Examples.* In Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, pp 45-52 (1992).

[9] Oncina, J. and García, P. *Inferring Regular Languages in Polynomial Updated Time.* In Pattern Recognition and Image Analysis. Pérez de la Blanca, Sanfeliú and Vidal (Eds.) World Scientific (1992).

[10] Trakhtenbrot B. and Barzdin Y. *Finite Automata: Behavior and Synthesis.* North Holland Publishing Company (1973).

Un Nuevo Modelo de Detector de Colisiones Basado en Elipsoides

Olmedo Arcila Guzmán, José María Bañón
Universidad del Valle, Escuela de Ingeniería de Sistemas y Computación
Cali, Colombia
olarcila, banon@eisc.univalle.edu.co

Abstract

In the collision detection literature, the bounding volumes are almost always simple primitives enclosing the boundary of the objects. This paper presents a collision detection model for convex polihedral objects based on the optimal enclosing and enclosed ellipsoids. While the enclosing ellipsoids are using to determine lack of intersection, the enclosed ellipsoids are used for detecting existence of intersection. In case of indetermination, we apply tests of intersection between pair of faces of the polyhedra objects. This system has been implemented and various numerical experiments were conducted. The results show that the use of enclosed ellipsoids increases significantly the efficiency of the detection.

Keywords: Collision Detection, Exact Algorithm, Ellipsoide, Bounding Volumes, Computational Geometry.

Resumen

En la literatura sobre la detección de colisiones los volúmenes limitantes son casi siempre primitivas sencillas que envuelven la frontera de los objetos. Este trabajo presenta un modelo de detección de colisiones para objetos poliedros convexos basado en los elipsoides óptimos envolventes y embebidos. Mientras que los elipsoides envolventes se usan para determinar la ausencia de colisión, los elipsoides embebidos son utilizados para detectar la existencia de colisión. En caso de indeterminación, se aplican tests de intersección entre pares de caras de poliedros. Este sistema ha sido implementado y diversos experimentos numéricos han sido realizados. Los resultados muestran que el uso de los elipsoides internos contribuye a incrementar considerablemente la eficiencia del detector.

Palabras claves: Detección de Colisiones, Algoritmo Exacto, Elipsoide, Volúmenes Limitantes, Geometría Computacional.

1. Introducción

En la literatura [5, 6, 8, 9, 13, 14, 16, 23] sobre detectores de colisiones de poliedros convexos una gran parte de los trabajos se centran en encontrar una buena jerarquía de volúmenes limitantes que ajuste adecuadamente a los objetos en varios niveles de precisión; normalmente se utilizan esferas ó cajas rectangulares [4]. Sin embargo, pocos autores consideran la representación elipsoidal, la cual, en general, representa a un poliedro convexo mejor que una esfera ó una caja rectangular. En este artículo se presenta el desarrollo de un detector de colisiones para poliedros convexos basado en el elipsoide mínimo que envuelve al poliedro y en el elipsoide máximo contenido en el poliedro. El detector utiliza la doble representación elipsoidal para acelerar los tests de ausencia y existencia de colisión y en caso de indeterminación procede al cálculo de la intersección entre las caras de los objetos. El algoritmo propuesto es implementado y se realizan varios experimentos y pruebas para estudiar su desempeño. En particular, se estudia la influencia del elipsoide interior en la detección temprana de colisiones y se muestra que tiene un efecto significativo en la reducción del tiempo de ejecución de la detección de la colisión. La organización del trabajo es la siguiente. La sección 2 describe los antecedentes del trabajo. El algoritmo del detector de colisiones se presenta en la sección 3. En la sección 4 se describen varios aspectos de la implementación del algoritmo. En la sección 5 se muestran varias pruebas realizadas con el detector de colisiones y se dan los resultados. Finalmente, en la sección 6 se dan las conclusiones y los futuros desarrollos del trabajo.

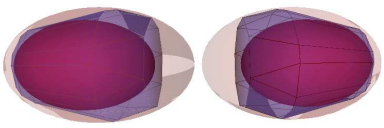
2. Antecedentes

La esfera [3] es un buen volumen limitante para representar objetos en la detección de colisiones por su sencillez geométrica. En efecto, basta comparar la distancia entre los centros de dos esferas con la suma de sus radios para saber si se intersectan. Por otra parte, debido a su simetría rotacional, la actualización de un objeto representado por esferas es muy sencilla pues la esfera depende de un centro y de un radio, es decir de cuatro parámetros. Sin embargo su principal desventaja es que es muy difícil conseguir con pocas esferas un ajuste preciso [7]. Ello es debido a que cuando un objeto es alargado la esfera que lo envuelve presenta mucho espacio inutil. Por ello varios autores [20, 21] han introducido los elipsoides para representar poliedros convexos debido esencialmente a que un elipsoide ajusta mejor, tanto por dentro como por fuera, un poliedro convexo que una esfera ó una caja rectangular. La representación elipsoidal es una representación sencilla de poliedros convexos que reduce la complejidad de la detección de colisiones. Hasta ahora la representación elipsoidal ha sido muy poco utilizada debido esencialmente a: *i*) la dificultad de calcular los elipsoides máximos y mínimos de un poliedro y *ii*) la dificultad de detectar la colisión entre dos elipsoides. Sin embargo, actualmente, el cálculo de los elipsoides máximos y mínimos de un poliedro es un problema de optimización que se resuelve sin dificultad con las actuales técnicas de optimización convexa. Además, existen técnicas algebraicas sencillas [21] que permiten decidir si existe colisión ó no entre dos elipsoides cualesquiera.
Rimon y Boyd [15] demuestran la utilidad de la representación elipsoidal en el contexto de la robótica en donde la mayor parte de las articulaciones de los robots son objetos alargados y por tanto pueden ser bien aproximados por elipsoides. En particular, estos autores envuelven tanto al robot como a los obstáculos con elipsoides y calculan la función claridad de las articulaciones con respecto a los obstáculos para ser utilizada en la detección de colisiones. En [10] y [12] se utilizan los elipsoides como volúmenes limitantes exteriores para ser utilizados en los detectores de colisión entre poliedros convexos. En [11] y [2] se presenta un modelo rápido de detección de colisiones y del cálculo de la distancia de separación para robots manipuladores basado en la utilización de los siguientes volúmenes limitantes para cada parte convexa del robot: *i*) el elipsoide máximo contenido en el objeto y *ii*) el elipsoide mínimo que contiene al objeto. Tal modelo es propuesto para la planificación interactiva del movimiento del robot en un simulador gráfico. La ventaja de la doble representación elipsoidal para poliedros convexos frente a otros volúmenes limitantes basados en cajas jerárquicas y en esferas es que reduce la complejidad y eficiencia del detector de colisiones.

3. El Detector de Colisiones

El detector de colisiones desarrollado está basado en la siguiente doble representación elipsoidal de poliedros convexos. Un poliedro convexo cualquiera $\mathcal{P}$ se representa por *i*) el elipsoide de volumen mínimo $\mathcal{E}_{\mathbf{min}}(\mathcal{P})$ que contiene a $\mathcal{P}$, y *ii*) el elipsoide de volumen máximo $\mathcal{E}_{\mathbf{max}}(\mathcal{P})$ contenido en $\mathcal{P}$. El elipsoide es una primitiva geométrica relativamente simple, descrita por una desigualdad cuadrática y especificada por nueve parámetros- tres para el centro, tres para las longitudes de sus ejes y otros tres para la orientación.
La estructura del algoritmo de detección de colisiones es bastante sencilla, se utilizan: *i*) tests de intersección entre elipsoides mínimos como test de ausencia de colisión, *ii*) tests de intersección entre elipsoides máximos como test de presencia de colisión y *iii*) si ambos fallan se calcula la colisión de forma exacta buscándola entre pares de caras de los cuerpos. Como entrada se tienen dos poliedros convexos $\mathcal{P}_\mathcal{A}$ y $\mathcal{P}_\mathcal{B}$, cuya colisión se requiere averiguar y el algoritmo debe finalmente reportar si existe colisión ó por el contrario indicar la ausencia de colisión. Consideramos los elipsoides mínimos $\mathcal{E}_{\mathbf{min}}(\mathcal{P}_\mathcal{A})$ y $\mathcal{E}_{\mathbf{min}}(\mathcal{P}_\mathcal{B})$, y los elipsoides máximos $\mathcal{E}_{\mathbf{max}}(\mathcal{P}_\mathcal{A})$ y $\mathcal{E}_{\mathbf{max}}(\mathcal{P}_\mathcal{B})$ de los poliedros. Primeramente, el algoritmo busca saber si hay ausencia de colisión entre los poliedros; para ello calcula si los elipsoides mínimos $\mathcal{E}_{\mathbf{min}}(\mathcal{P}_\mathcal{A})$ y $\mathcal{E}_{\mathbf{min}}(\mathcal{P}_\mathcal{B})$ se tocan ó no se tocan. Si ambos elipsoides no se tocan evidentemente tampoco se tocan los poliedros contenidos y el algoritmo finaliza reportando la ausencia de colisión. De lo contrario el algoritmo busca saber si hay colisión entre los poliedros, para ello calcula si los elipsoides máximos $\mathcal{E}_{\mathbf{max}}(\mathcal{P}_\mathcal{A})$ y $\mathcal{E}_{\mathbf{max}}(\mathcal{P}_\mathcal{B})$ se tocan ó no se tocan. Si ambos elipsoides se tocan entonces hay también colisión entre los poliedros respectivos y el algoritmo finaliza reportanto la colisión. En el caso en que los elipoides máximos no se toquen entonces nada se puede concluir. En este caso el algoritmo procede a calcular exactamente si existe intersección entre cualquier par de caras poligonales de los poliedros. Ello se realiza triangularizando las caras poligonales de cada poliedro y utilizando un test de intersección entre triángulos en el espacio. Si algún triángulo de las caras de un poliedro colisiona con un triángulo de las caras del otro poliedro entonces el algoritmo reporta colisión, de lo contrario reporta su

ausencia. El algoritmo dearrollado es pues exacto.
Con respecto al test de intersección entre elipsoides cualesquiera se ha utilizado un método algebraico desarrollado en [21] que permite resolver analíticamente si dos elipsoides cualesquiera intersecionan ó no.



(a) Los Elipsoides Externos no se Solapan

(b) Los Elipsoides Internos se Solapan

(c) Solapamiento entre las Caras de los Poliedros

Figura 1: Proceso de Detección de Intersección.

La figura 1 ilustra gráficamente el funcionamiento del detector de colisiones. Se muestran dos poliedros convexos en forma de diamante (poliedro C, ver *Figura 2.c*) con sus respectivos elipsoides máximos y mínimos. En la figura 1.a los elipsoides externos no se solapan, los poliedros no colisionan. En la figura 1.b los elipsoides internos se solapan, hay colisión entre los poliedros. En la figura 1.c los elipsoides externos se intersectan pero los internos no, la detección de colisión se realiza mediante tests de intersección entre las caras de los poliedros.

4. Implementación

En esta sección se describen varios conceptos, definiciones, teoremas y fórmulas que se han utilizado en la implementación del detector de colisiones.
El problema del cálculo de los elipsoides de volumenes máximos y mínimos de un poliedro convexo cualquiera es un problema típico de optimización convexa [1, 17, 18, 19]. Debido a que el volumen de un elipsoide es proporcional al determinante de una matriz, este problema cae dentro de los problemas de optimización denominados **maxdet**, que consisten en el cálculo del máximo del determinante de una matriz sujeto a desigualdades matriciales: minimizar $\log\det G(x)^{-1}$, tal que $G(x) > 0$ y $F(x) \geq 0$, donde la variable de optimización es el vector $\mathbf{x} \in \mathbb{R}^m$. Las matrices $G : \mathbb{R}^m \mapsto \mathbb{R}^{l\times l}$ y $F : \mathbb{R}^m \mapsto \mathbb{R}^{n\times n}$ son transformaciones afines.
En este trabajo se ha utilizado el programa SDPSOL [19, 22] para resolver los problemas de optimización de elipsoides. SDPSOL es un programa que resuelve los problemas de maximización de determinantes y problemas de programación semidefinida mediante el algoritmo del punto interior [18, 19]. La ventaja de la utilización de SDPSOL es que permite la especificación de problemas en función de variables de optimización que presentan estructura matricial y de desigualdades matriciales que aparecen en el planteamiento del problema.

4.1. Cálculo del Elipsoide de Volumen Mínimo

El problema del cálculo del elipsoide mínimo es el de dado un poliedro convexo $\mathcal{P}$ con vértices $\{\mathbf{x_i}\}_i$, encontrar el elipsoide de volumen mínimo que contiene a $\mathcal{P}$. Para plantear el problema conviene tomar la

siguiente definición del elipsoide,

$$\{\mathbf{x} \in \mathbb{R}^3 : \|E\mathbf{x} + \mathbf{c}\| \leq 1\},$$

es decir, un elipsoide es la imagen inversa de la esfera unidad de una transformación afin E. Puede suponerse, sin pérdida de generalidad, que $E = E^t > 0$. Se demuestra que el volumen del elipsoide es proporcional al inverso del determinante de la matriz E. El problema puede plantearse matemáticamente como el siguiente problema de optimización **maxdet**:

$$\begin{array}{ll} \text{minimizar} & \log\det E^{-1} \\ \text{tal que} & \|E\mathbf{x_i} + \mathbf{c}\| \leq 1, \\ & E = E^t > 0 \end{array} \tag{1}$$

La restricción de la norma $\|E\mathbf{x_i} + \mathbf{c}\| \leq 1$ es una desigualdad convexa cuadrática en las variables E y $\mathbf{c}$. Puede ponerse como una desigualdad matricial de la forma siguiente:

$$\begin{bmatrix} I & E\mathbf{x_i} + \mathbf{c} \\ (E\mathbf{x_i} + \mathbf{c}) & 1 \end{bmatrix} \geq 0. \tag{2}$$

Una vez que se tiene el problema formulado en función de desigualdades matriciales se utiliza SDPSOL para calcular el elipsoide mínimo correspondiente.

4.2. Cálculo del Elipsoide de Volumen Máximo

El problema del cálculo del elipsoide máximo es el de dado un poliedro convexo cualquiera $\mathcal{P}$ encontrar el elipsoide $\mathcal{E}$ de volumen máximo contenido en $\mathcal{P}$. Un poliedro convexo cualquiera $\mathcal{P}$ está definido por un conjunto de desigualdades siguientes,

$$\mathcal{P} = \{\mathbf{v} \in \mathbb{R}^3 : M\mathbf{v} \leq b\}$$

donde $M \in \mathbb{R}^{m\times 3}, m > 3$, es una matriz de rango 3. Las desigualdades anteriores expresan el hecho de que todo poliedro convexo es la intersección de un número finito de semiplanos. Para plantear analíticamente el problema, conviene considerar que un elipsoide es la imagen de la esfera unidad mediante una transformación afin E, es decir

$$\mathcal{E} = \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x} = \mathbf{c} + E\mathbf{s}, \|s\| = 1\}$$

Sin pérdida de generalidad se puede suponer que $E = E^t > 0$. En este caso se demuestra que el volumen del elipsoide definido es proporcional al determinante de E. Entonces, el problema del elipsoide máximo puede plantearse matemáticamente como el siguiente problema de optimización maxdet:

$$\begin{array}{ll} \text{maximizar} & \log\det E \\ \text{tal que} & \mathbf{c} + E\mathbf{s} \in \mathcal{P}, \forall\|s\| = 1, \\ & E = E^t > 0 \end{array} \tag{3}$$

donde la matriz E y el centro del elipsoide $\mathbf{c}$ son las variables. Este problema puede ponerse bajo la forma de la siguiente desigualdad matricial,

$$\begin{bmatrix} (b_i - m_i^t\mathbf{c})I & Em_i \\ m_i^tE & b_i - m_i^t\mathbf{c} \end{bmatrix} \geq 0. \tag{4}$$

Análogamente, una vez que se tiene el problema formulado en función de desigualdades matriciales se utiliza SDPSOL para calcular el elipsoide máximo correspondiente.

4.3. Test de Intersección entre Elipsoides

Para detectar si dos elipsoides cualesquiera colisionan se ha utilizado la condición de separación de dos elipsoides desarrollada en [21] que permite resolver analíticamente si dos elipsoides intersectan ó no.
Se denomina polinomio característico $f(\lambda)$ del sistema formado por los elipsoides definidos por las matrices en coordenadas homogéneas $\mathcal{A}$ y $\mathcal{B}$, al polinomio definido por la siguiente expresión,

$$f(\lambda) = \det(\lambda\mathcal{A} + \mathcal{B})$$

En términos de las raices del polinomio característico del sistema existen condiciones necesarias y suficientes para saber cuando ambos elipsoides estan separados ó se tocan en un solo punto. La ecuación $f(\lambda) = 0$ se denomina ecuación característica.

Teorema: Sean $\mathcal{A}$ y $\mathcal{B}$ dos elipsoides cualesquiera con la ecuación característica $f(\lambda) = 0$. Entoces, los dos elipsoides estan separados sí y solo sí la ecuación característica tiene dos raices distintas positivas. Los elipsoides se tocan externamente en un punto sí y solo sí la ecuación característica tiene una raiz doble positiva.

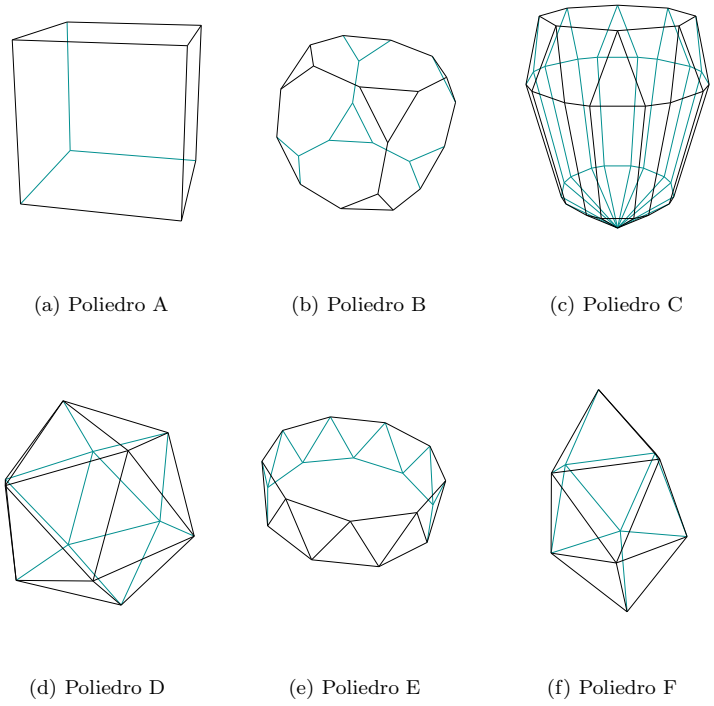
5. Experimentación y Discusión

Con el fin de estudiar el desempeño del detector de colisiones implementado se han realizado varios experimentos. Los cuerpos utilizados en los experimentos son los seis poliedros convexos que se describen en el cuadro 1 y se muestran en la figura 2.

Cuerpos	Vértices	Caras
Poliedro A	8	6
Poliedro B	24	14
Poliedro C	41	49
Poliedro D	12	20
Poliedro E	18	20
Poliedro F	20	16

Cuadro 1: Poliedros utilizados en los experimentos de prueba del algoritmo.

Se ha utilizado la herramienta de optimización SDPSOL para calcular los elipsoides mínimo y máximo de cada poliedro. Este último cálculo es considerado como un preprocesamiento; es decir los elipsoides mínimo y máximo se calculan una sola vez al principio para una posición inicial del poliedro. Durante el experimento a medida que las posiciones y orientaciones de los poliedros cambian se actualizan las correspondientes de los elipsoides.
Para proceder a la detección de colisiones entre pares de poliedros se procede en cada ejecución: *i*) a colocarlos mediantes traslaciones y rotaciones aleatorias en el interior de una caja rectangular, *ii*) seguidamente se calcula la detección de colisiones y *iii*) estos dos procesos se repiten 100000 veces y se promedian los tiempos obtenidos. Todo el proceso anterior se repite 10 veces para obtener una mejor estadística. El tamaño de la caja rectangular es importante pues si es muy grande respecto al tamaño de los poliedros entonces ocurren pocas colisiones en comparación al número de no colisiones. Por ello, en los distintos experimentos realizados se han tomado los tamaños de la caja rectangular de manera que aproximadamente ocurran tantas colisiones como no colisiones. La ejecución ha sido realizada en una máquina DELL PRECISION 530, con un procesador Intel-Xeon con 1GB de RAM, sobre Debian GNU/Linux 3.0 usando el compilador C++/GNU g++ versión 2.95.4.
En el cuadro 2 se muestran algunos de los resultados obtenidos de los tiempos de la detección de una colisión $T_{\mathbf{Int}.}$ y de los tiempos de la detección de una no-colisión $T_{\mathbf{NOInt}.}$ en microsegundos para ciertos pares de poliedros. Tales resultados corresponden a dos tipos diferentes de experimentos. Las columnas segunda y tercera de el cuadro 2 corresponden a la ejecución del detector de colisiones sin elipsoides internos, es decir la detección de colisión ocurre mediante los tests de intersección entre las caras de los poliedros. Las columnas cuarta y quinta del mismo cuadro corresponden a la ejecución del detector de colisiones con elipsoides



(a) Poliedro A (b) Poliedro B (c) Poliedro C

(d) Poliedro D (e) Poliedro E (f) Poliedro F

Figura 2: Poliedros Utilizados.

internos. Comparando el tiempo de la detección de una colisión $T_{\mathbf{Int.}}$ en ambos experimentos notamos que este es significativamente inferior cuando se tienen en cuenta los elipsoides internos. Evidentemente, ello es debido a que los elipsoides internos contribuyen a la detección temprana de la colisión. Como se muestra en el cuadro 2, cuando se tiene en cuenta los elipsoides internos la reducción en el tiempo $T_{\mathbf{Int.}}$ puede llegar al 50 %. Los tiempos de la detección de una no-colisión, columnas tercera y quinta, son prácticamente los mismos siendo la diferencia entre ellos de naturaleza estadística. Ello es debido a que la detección de la ausencia de colisión es debida a los elipsoides exteriores que son considerados en ambos experimentos.
En el caso de la detección de colisiones con elipsoides internos los tiempos de detección de una colisión que aparecen en el cuadro 2 dependen del tamaño del entorno de trabajo, es decir del tamaño de la caja rectangular en la cual se colocan los pares de poliedros. Si los poliedros están en un espacio de trabajo reducido entonces la detección de colisiones va ser debida predominantemente a los elipsoides internos. Si el espacio de trabajo es grande con respecto al tamaño de los poliedros entonces tanto la contribución de los los elipsoides internos como la de los tests de intersección entre pares de caras de poliedros es importante en la detección de colisiones. Con el objeto de cuantificar los tiempos de la detección de una colisión mediante *i*) los elipsoides internos, y *ii*) los tests de intersección entre caras, se han realizado medidas de estos tiempos en los experimentos realizados. El cuadro 3 resume el resultado de tales medidas. En el cuadro 3 se indican: *i*) el tiempo de detección de una colisión en el experimento, *ii*) el porcentaje de las detecciones debidas a los elipsoides internos y el tiempo promedio de cada una de ellas, y *iii*) el porcentaje de las detecciones debidas al test de intersección entre caras y el tiempo promedio de cada una de ellas. Como se muestra en el cuadro 3 el tiempo de la detección de una colisión mediante los elipsoides internos es un orden de magnitud inferior al tiempos de la detección de una colisión mediante los tests de intersección entre caras. La detección de una colisión mediante los elipsoides internos es un procedimiento rápido con respecto a la verificación de intersección entre pares de caras de poliedros. Ello explica el porqué los elipsoides internos son tan eficientes en la detección de colisiones; contribuyen a la disminución del tiempo de la detección de la colisión.

	Sin Elipsoides Internos		*Con Elipsoides Internos*	
Poliedros	$T_{\mathbf{Int.}}$ (μs)	$T_{\mathbf{NOInt.}}$ (μs)	$T_{\mathbf{Int.}}$ (μs)	$T_{\mathbf{NOInt.}}$ (μs)
A - B	111.95	422.66	82.84	426.73
A - C	320.07	669.85	165.64	682.12
A - D	126.81	39.06	68.17	39.29
A - E	87.83	335.60	58.74	342.77
A - F	75.49	161.56	58.66	162.65
B - C	593.44	980.44	289.21	991.66
B - E	275.7	496.08	149.94	493.76
B - F	184.06	314.38	91.49	325.43

Cuadro 2: Impacto de Elipsoides Internos en la Obtención de Tiempos de Detección de Colisiones.

		Elipsoides Internos		*Tests Caras*	
Poliedros	$T_{\mathbf{Int.}}$	%	T (μs)	%	T (μs)
A - B	82.84	56.02 %	27.72	43.98 %	153.08
A - C	165.64	61.11 %	27.56	38.89 %	382.66
A - D	68.17	71.22 %	26.89	28.78 %	170.35
A - E	58.74	61.95 %	17.20	38.05 %	126.39
A - F	58.66	59.38 %	15.88	40.62 %	121.18
B - C	289.21	62.67 %	14.99	37.33 %	749.67
B - E	149.94	53.24&	16.43	46.76 %	379.71
B - F	91.49	64.14 %	15.72	35.86 %	227.04

Cuadro 3: Comparación entre las Intersecciones Detectadas por los Elipsoides Internos y las Caras del Cuerpo.

6. Conclusiones y Futuros Desarrollos

En este artículo se ha desarrollado e implementado un detector de colisiones para poliedros convexos basado en los elipsoides externo máximo e interno mínimo de un poliedro. Los elipsoides máximo y mínimo han sido calculados a partir de eficientes técnicas de programación convexa. El detector de colisiones desarrollado es exacto; utiliza los test de intersección entre los elipsoides externos para verificar la ausencia de colisiones, los test de intersección entre elipsoides internos para detectar tempranamente la colisión, y en caso de indeterminación se realiza tests de intersección entre pares de caras de poliedros.
La experimentación realizada con el detector demuestra cuantitativamente que los tiempos de la deteción de colisiones utilizando elipsoides internos son considerablemente inferiores que los obtenidos sin utilizar elipsoides internos. Ello es debido a que los elipsoides internos permiten detectar tempranamente y rápidamente la detección de colisión.
Se debe notar que los sistemas más importantes de detección de colisiones (AABB, OBB, etc.) no consideran volúmenes limitantes internos para la detección de colisiones. Este artículo es el primer trabajo que muestra cuantitativamente la importancia de considerar volúmenes limitantes internos en el diseño de detectores de colisión. Aunque el detector de colisiones desarrollado está basado en elipsoides los resultados obtenidos en este trabajo son generales y aplicables a otros detectores de colisión. Se afirma que si se consideran volúmenes limitantes internos en los detectores de colisión que trabajan con jerarquías de volumenes limitantes externos, se puede mejorar considerablemente sus desempeños. Se necesita más experimentación en la aplicación de la doble representación elipsoidal para comparar su desempeño con respecto a otros métodos.
Como desarrollo futuro se trabaja en el diseño e implementación de modelos de detectores de colisión basados en jerarquías de cajas rectangulares externas e internas y con jerarquías de esferas externas e internas. Asímismo se trabaja en comparar el desempeño del detector de colisiones desarrollado con otros detectores.

Referencias

[1] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[2] Y. R. Chien and J. S. Liu. Improvements to ellipsoidal fit based collision detection. Technical report, Institute of Information Science 20, Academia Sinica. tR03001.pdf, 2003.

[3] A. P. del Pobil and M. A. Serna. A new representation for robotics and artificial intelligence applications. *International Journal of Robotics and Automation*, 9(1):11–21, 1994.

[4] A. P. del Pobil and M. A. Serna. *Spatial Representation and Motion Planning*. Springer-Verlag, 1995.

[5] S. Gottschalk. RAPID: Robust an Accurate Polygon Interference Detection System, 1996.

[6] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. *Computer Graphics*, 30(Annual Conference Series):171–180, 1996.

[7] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[8] T. C. Hudson, M. C. Lin, J. Cohen, S. Gottschalk, and D. Manocha. V-COLLIDE: Accelerated Collision Detection for VRML. In *VRML '97: Proceedings of the second symposium on Virtual reality modeling language*, pages 117–ff. ACM Press, 1997.

[9] T. Klosowski J, Held M, S. B. Mitchell J, Sowizral H, and Zikan K. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[10] M. Y. Ju, J. S. Liu, Y. R. Chien, K. S. Hwang, and C. Lee. A Novel Collision Detection Method Based on Enclosed Ellipsoid. In *2001 IEEE Interantional Conference of Robotics and Automation*, pages 21–26, 2001.

[11] M. Y. Ju, J. S. Liu, and K. S. Hwang. Ellipsoidal modeling for articulated robot manipulators for interactive motion planning. Technical report, Institute of Information Science 20, Academia Sinica. TR-IIS-00-008, 200.

[12] M. Y. Ju, J. S. Liu, S. P. Shiang, Y. R. Chien, K. S. Hwang, and W. C. Lee. Fast and accurate collision detection based on enclosed ellipsoid. In *Proceedings of the Sixth International Conference on Control*, volume 19, pages 381–394. Robotics and Vision (ICARCV2000) TM7.6, 2000.

[13] B. Mirtich. V-Clip: Fast and Robuts Polyhedral Collision Detection. *ACM Transaccions on Graphics*, 17(3):177–208, July 1998.

[14] C. O'Sullivan and J. Dingliana. Real-time collision detection and response using sphere-trees, 1999.

[15] E. Rimon and S. Boyd. Efficient distance computation using best ellipsoid fit. Technical report, Stanford University, Department of Electrical Engineering, 1992.

[16] G. van den Bergen. Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools: JGT*, 2(4):1–14, 1997.

[17] L. Vandenberghe and S. Boyd. Semidefinite Programming. *SIAM Review*, 38(1):49–95, 1996.

[18] L. Vandenberghe and S. Boyd. Connections Between Semi-infinite and Semidefinite Programming. In *Semi-infinite Programming*, pages 277–294. Kluwer Acad. Publ., Boston, MA, 1998.

[19] L. Vandenberghe, S. Boyd, and S.-P. Wu. Determinant Maximization with Linear Matrix Inequality Constraints. *SIAM J. Matrix Anal. Appl.*, 19(2):499–533, 1998.

[20] W. Wang, Y. Choi, B. Chan, M. Kim, and J. Wang. Efficient Collision Detection for Moving Ellipsoids Bases on Simple Algebraic Test and Separating Planes. Technical report, The University of Hong Kong, Hong Kong., Seoul National University, Seoul, South Korea., Shandong University, Jinan, China., 2002.

[21] W. Wang, J. Wang, and M. Kim. An Algebraic Condition for the Separation of Two Ellipsoids. In *Computer Aided Geometric*, volume 18, pages 531–539. Elsevier Science B.V., 2001.

[22] S.-P. Wu and S. Boyd. SDPSOL: A parser/solver for SDP and MAXDET problems with matrix structure, June 1996.

[23] G. Zachmann. The BoxTree: Exact and Fast Collision Detection of Arbitrary Polyhedra SIVE, 1995.

Apéndice

La superficie de un elipsoide en en coordenadas cartesianas de $\mathbb{R}^3$ está definido como el conjunto de puntos $\mathbf{x} \in \mathbb{R}^3$ tales que,

$$\{\mathbf{x} \in \mathbb{R}^3 : (\mathbf{x} - \mathbf{c})^t A(\mathbf{x} - \mathbf{c}) = 1\}$$

donde A es una matrix de dimensión 3×3 que define el elipsoide y $\mathbf{c}$ es su centro. La matriz A es una matriz simétrica y definida positiva. Sin perder generalidad se puede afirmar que $A = (EE^t)^{-1}$ donde la matriz E es también simétrica y definida positiva.
Una forma equivalente de definir el elipsoide es:

$$\{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x} = \mathbf{c} + E\mathbf{s}, \|s\| = 1\}$$

Tomando $B = E^{-1}$ y $\mathbf{b} = -E^{-1}\mathbf{c}$, y sustituyéndolos en la ecuación anterior se llega a otra definición equivalente de elipsoide,

$$\{\mathbf{x} \in \mathbb{R}^3 : \|B\mathbf{x} + \mathbf{b}\| \leq 1\}.$$

En coordenadas homogéneas $X = (x, y, x, w)^t$ la ecuación de un elipsoide se expresa como el conjunto de puntos $X \in \mathbb{R}^4$ tales que.

$$\{X \in \mathbb{R}^4 : X^t \mathcal{A} X = 0\}$$

Método de Visualización de Volúmenes sobre Superficies VoS

Ms. Patrícia S. Herrera Cateriano
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, SP, Brasil
patricia@icmc.usp.br

and

Dr. Luis Gustavo Nonato
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, SP, Brasil
gnonato@icmc.usp.br

Abstract

This paper presents a new method for visualization that take advantage of the direct volume rendering and surface rendering in a hybrid environment. This method, named VoS, uses a pre-visualization over the bounding of the volume that make possible a real time interaction with the volumetric objects modeled as unstructured meshes. Also, this new method of visualization is parallelizable and can be accelerated with common graphical cards.

Keywords: Computer graphics, volume visualization, hybrid volume rendering, unstructured meshes.

Resumen

Este trabajo presenta un nuevo método de visualización que aprovecha las ventajas del *rendering* volumétrico directo y del *rendering* de superficies en un ambiente híbrido. El método, denominado VoS, utiliza una pre-visualización sobre el borde del volumen que viabiliza una interacción en tiempo real con objetos volumétricos modelados por medio de mallas no estructuradas. Así también, este nuevo método de visualización es paralelisable y puede ser acelerado con placas gráficas comunes.

Palabras claves: Computación gráfica, visualización volumétrica, *rendering* volumétrico híbrido, mallas no estructuradas.

1. INTRODUCCIÓN

El gran adelanto tecnológico de los últimos años propició el crecimiento de aplicaciones que producen datos volumétricos. Algunos ejemplos son las tomografías por computadora, la microscopía y las mediciones sísmicas. La cantidad de información generada por estas aplicaciones es inmensa y necesita ser analizada e interpretada de forma simple, clara y rápida. Para hacer mas ágil este proceso científico de descubrimiento, confirmación y reproducción de datos son utilizadas herramientas como los métodos de Visualización Volumétrica.

Típicamente, las técnicas de Visualización Volumétrica son divididas en dos grupos: las técnicas de *Rendering* de Superficies (RS) y las técnicas de *Rendering* Volumétrico Directo (RVD). Las técnicas de RS utilizan aproximaciones poliedrales basadas en primitivas geométricas. Las técnicas de RVD se popularizaron

por presentar imágenes realistas y de alta calidad sin utilizar representaciones geométricas intermediarias, y que además conservan la noción global de las características de los datos. Muchas aplicaciones se basaron en un grupo u otro, sin embargo, en muchas aplicaciones es necesario no apenas visualizar los datos que definen el volumen, sino también modelar estructuras contenidas en el interior de tales volúmenes. Por eso, es bastante útil el desarrollo de algoritmos híbridos que permitan la visualización simultánea de datos volumétricos y estructuras poligonales modeladas a partir de esos datos.

Un otro aspecto de las técnicas de Visualización Volumétrica es la representación de los datos. Las técnicas deben sacar provecho de la representación utilizada para la manipulación de datos. Así, la cuestión de los modelos de representación (*voxels* o células) y del almacenamiento de los datos es otro factor importante durante la visualización.

Los diferentes aspectos mencionados aquí motivaron el desarrollo de una solución alternativa para el *rendering* volumétrico de modo que sea fácil la obtención de interacción flexible y rápida para mallas no estructuradas. La técnica desarrollada recibió el nombre de VoS (Volume Visualization on Surfaces). Esta técnica es una solución híbrida que utiliza los métodos de Rendering Volumétrico Directo y de Rendering Superficial con la finalidad de posibilitar una visualización en tiempo real.

Este artículo describe en términos generales los conceptos y procesos en los cuales se basa la solución híbrida propuesta. En la sección 2 son presentados algunos conceptos generales de la Visualización Volumétrica. Luego en la sección 3 son introducidas las técnicas de Visualización Híbrida. En la sección 4 es presentada la Técnica Híbrida VoS desarrollada durante nuestras investigaciones. La sección 5 menciona algunos detalles de la implementación de la técnica VoS. Finalmente, en la sección 6 son presentadas algunas conclusiones sobre la investigación.

2. VISUALIZACIÓN VOLUMÉTRICA

Visualización Volumétrica es el proceso de proyectar un conjunto de datos multidimensionales en una imagen bidimensional plana [El92]. El objetivo de la Visualización Volumétrica es proveer mecanismos para visualizar el contenido de los conjuntos de datos volumétricos y manipular estas estructuras volumétricas, complejas y dinámicas. La Visualización Volumétrica investiga estructuras para representar los datos y técnicas de *rendering* para su visualización.

Frecuentemente, el conjunto de datos a ser visualizado es definido como una malla tridimensional, con un o más valores escalares y, posiblemente, un o más valores vectoriales en cada elemento de la malla. La forma de representación de estas mallas varía desde mallas regulares, típicamente formadas por unidades de volumen cúbicas denominadas *voxels*, hasta mallas irregulares, formadas por células que presentan características más generales que los *voxels*. Tales células pueden adoptar diferentes formas geométricas, entre ellas la tetraedral, que es una de las más utilizadas actualmente.

Los motivos que llevan al uso de elementos tetraedrales en la descomposición de volúmenes son: la mayor parte de las células pueden ser descompuestas en tetraedros; el *rendering* puede ser calculado más fácilmente para mallas tetraedrales; la extracción de iso-superficies es más robusta pues no presenta ambigüedades que aparecen con otras representaciones; la mayoría de los algoritmos de *rendering* son más simples y rápidos de describir e implementar para mallas tetraedrales.

Además de las ventajas mencionadas antes, las mallas tetraedrales son adecuadas para modelar datos en cualquier dimensión; son bases adecuadas para muchas técnicas de interpolación; el proyecto de estructuras de datos es simplificado; y el tratamiento de casos especiales es más simple [CMS97].

2.1. Clasificación de los Algoritmos de Visualización Volumétrica

Los algoritmos fundamentales de Visualización Volumétrica, según las técnicas de *rendering* que utilizan, pertenecen a dos categorías básicas:

1. *Rendering* de Superficies (RS) - *Surface Rendering.*

2. *Rendering* Volumétrico Directo (RVD) - *Direct volumen Rendering*.

El *Rendering* de Superficies, también conocido como reconstrucción de superficies mediante algoritmos, es generalmente hecho a través de aproximaciones poliedrales de iso-superficies extraídas del volumen. El *Rendering* Volumétrico Directo utiliza la proyección directa de los datos que componen el volumen para generar una visualización.

2.1.1. Rendering de Superficies (RS)

Los algoritmos de RS generalmente utilizan primitivas geométricas para generar un modelo del objeto, a partir del cual, el volumen fue obtenido. En este caso, son generadas apenas las superficies que limitan regiones de los objetos, siendo que las técnicas más comunes son: extracción de iso-superficies [LC87, CLLC88, Le90b, HL79, CMS97, CMPS97] y los métodos de reconstrucción a partir de contornos [EPO91, JC94, FKU97, Ka98, BCL99, KS00].

Entre las ventajas de los algoritmos de RS se observa que generalmente son más simples de implementar y más rápidos que los algoritmos RVD, pues el RS atraviesa el volumen una única vez para extraer las superficies.

El RS también presenta desventajas. Dado que es utilizada solamente una representación de la superficie, mucha información contenida en los datos es perdida durante el proceso de *rendering*. Algunos otros problemas son la introducción ocasional de falsos positivos (formas que no existen); o de falsos negativos (no llevar en cuenta pequeñas características de los datos o definidas de forma muy simple). Los falsos positivos y negativos pueden ser vistos incorrectamente por los usuarios como características de los datos [El92].

Las principales técnicas de extracción por iso-superficie para mallas regulares son: *Cuberille* [HL79] y *Marching Cubes* [CLLC88], y para mallas no estructuradas una de las principales técnicas desarrolladas fue una extensión del algoritmo *Marching Cubes* denominada *Marching Tetrahedra* [CMS97]. En la reconstrucción a partir de contornos encontramos como el algoritmo más representativo al *Contour-Connecting* [Ke95].

2.1.2. Algoritmos de Rendering Volumétrico Directo (RVD)

Los métodos de RVD son capaces de preservar las informaciones volumétricas en la visualización, hecho que ha llevado a muchos investigadores a invertir en este tipo de estrategia [NMS00]. Estas técnicas objetivan crear una imagen directamente a partir de los datos volumétricos, sin el auxilio de primitivas geométricas [Ka98]. Las técnicas de RVD se volvieron populares por evitar la necesidad de hacer un gran preprocesamiento de los datos y por ser relativamente fáciles de implementar para el caso de volúmenes regulares.

La forma más simples (e ineficiente) de implementar una técnica de RVD es utilizando la técnica *Z-Buffer*. La idea es atravesar el volumen entero considerando cada *voxel* como un punto 3D que es transformado, utilizando una matriz de proyección en un *Z-Buffer*, y diseñando en la pantalla. Esta técnica es una de las más simples y fue implementada para mallas regulares formadas por *voxels*. Posteriormente fueron desarrolladas técnicas de RVD más eficientes tanto para mallas regulares [KS86, HM90, Le90b] como para mallas irregulares [Ga90, RW92, YRLS96, CMS97, CTT00].

La gran ventaja de los algoritmos RVD es la alta calidad de la imagen producida. En contrapartida, la necesidad de atravesar el volumen diversas veces cuando una imagen es producida, hace al algoritmo RVD lento, computacionalmente caro, y la interacción del usuario con la imagen 3D producida es ineficiente, lo cual es un punto crítico en la visualización. Esos problemas han representado importantes desafíos para los investigadores que estudian esta técnica.

La representación de los volúmenes en el RVD es hecho utilizando mallas cartesianas, las cuales son la forma típica de representación, mas actualmente se busca utilizar una forma de representación más general, como son las mallas no estructuradas. Aquí, las operaciones de *rendering* presentan nuevos problemas y ambigüedades. En los diferentes abordajes referentes al *rendering* volumétrico de mallas irregulares, extender la operación de muestreo de puntos en el lanzamiento de rayos (*Ray Casting*) se presenta como un desafío para los investigadores. La cuestión de ordenación de las células tampoco es una operación trivial [Ga90].

3. TÉCNICAS DE VISUALIZACIÓN HÍBRIDAS

Muchas veces, la forma de visualizar datos volumétricos varia conforme los requisitos de los usuarios y la complejidad de los datos. Esto lleva a actualizaciones constantes de los algoritmos y técnicas de visualización volumétrica empleadas en el análisis y manipulación de tales datos. Esas actualizaciones, por su vez, buscan nuevas formas de optimización para los métodos de visualización, resultando en algoritmos híbridos, paralelismo, accesibilidad en redes, y visualización en tiempo real.

El término “algoritmo híbrido de visualización” es utilizado cuando se hace referencia a algoritmos que, de alguna forma, hacen uso de dos o más estrategias de visualización en un único contexto.

Gran parte de los algoritmos de visualización pueden ser vistos como optimizaciones a partir de una misma idea básica, que fue sufriendo alteraciones durante su evolución. Muchas veces estas optimizaciones fueron desarrolladas paralelamente, y posteriormente agrupadas. Muchos autores denominaron esos algoritmos de híbridos, pero se basan en un mismo tipo de *rendering* realizado en un mismo espacio de proyección**.** Consideramos todos estos algoritmos como optimizaciones de un mismo abordaje.

Después del levantamiento bibliográfico notamos que los algoritmos híbridos tienden a ser de tres tipos:

- Algoritmos híbridos basados en el tipo de *rendering;*
- Algoritmos híbridos basados en el dominio del volumen;
- Algoritmos híbridos basados en la arquitectura de computación.

Los algoritmos híbridos basados en el tipo de *rendering* abarcan aquellos algoritmos que combinan la visualización de un volumen de datos con la visualización de un conjunto de polígonos, intentando resolver problemas envolviendo no sólo la información del volumen de datos, mas también informaciones espaciales específicas, que sólo pueden ser proporcionadas por los objetos definidos geométricamente. Los algoritmos en esta categoría son muchos, dos ejemplos son presentados a seguir.

En uno de los trabajos de Levoy [Le90a] es presentado un algoritmo de *Ray Tracing* Híbrido que mixtura un volumen de datos con un conjunto de polígonos, de modo que aparezcan juntos en una misma imagen. La idea es que los rayos son lanzados simultáneamente en el conjunto de polígonos y en el conjunto de datos volumétricos. Muestras de cada uno son extraídas a intervalos igualmente espaciados en el camino de los rayos, y los colores y opacidades resultantes son compuestos, siguiendo el orden de la profundidad. Para evitar el efecto de *aliasing* en las aristas de los polígonos, fueron utilizadas formas de súper muestreo selectivo. Para evitar errores de visibilidad, en los puntos de intersección entre los polígonos y el volumen, fue dado un tratamiento especial para las muestras del volumen que se encontraron inmediatamente al frente o atrás de los polígonos.

Los algoritmos híbridos basados en el dominio del volumen hibridizan técnicas volumétricas basadas en diferentes espacios de visualización. Un ejemplo es el caso que combina técnicas de proyección a partir del dominio de los objetos con técnicas de proyección a partir del dominio de la imagen, un ejemplo característico es el propio algoritmo *Shear-Warp* [LL94], que transforma un volumen de datos para simplificar la etapa de proyección del *pipeline* de visualización. Esto es obtenido por medio de la aplicación de una translación en las capas (*slices*) del volumen, transformando los datos en un sistema intermedio de coordenadas, donde los rayos de visión son perpendiculares a los cortes del volumen generando imágenes distorsionadas que luego son corregidas para la visualización final.

También encontramos algoritmos que utilizan arquitecturas más especializadas para desarrollar técnicas tanto secuénciales [ST90, Wi92], cuanto paralelas [OUT85, JC94]. Los algoritmos de visualización volumétrica para arquitecturas especializadas, particularmente para arquitecturas paralelas son muy importantes porque esas arquitecturas soportan enormes cantidades de datos volumétricos, como en el caso de las simulaciones numéricas. En estos casos es muy inconveniente mover estos conjuntos de datos para *workstations* a fin de realizar una visualización con post-procesamiento.

Además de ser consideradas las múltiples formas de clasificación de las arquitecturas paralelas, se debe llevar en cuenta también el tipo de algoritmo de *rendering* a ser paralelisado. Una descripción apropiada de las arquitecturas paralelas y sus clasificaciones puede ser encontrada en [Ho93]. Una descripción de las principales técnicas de medida de desempeño, como la escalabilidad y su relación con la velocidad de procesamiento, puede ser encontrada en [HM90].

4. TÉCNICA HÍBRIDA DE VISUALIZACIÓN DE DATOS VOLUMÉTRICOS NO ESTRUCTURADOS

La técnica presentada aqui hibridiza técnicas de RS con RVD. Esta integración no es una tarea fácil, pero es una característica deseada en muchas aplicaciones**.** Asi, la técnica de *rendering* propuesta proporciona una pre-visualización directa en la superfície del volumen lo que permite acelerar las transformaciones de proyección, viabilizando una mayor interacción con el usuário.

Además, esta técnica híbrida utiliza una forma general de representación mediante mallas no estructuradas. Al utilizar este tipo de mallas, debemos considerar que, tanto la configuración geométrica de estas mallas, como los cálculos matemáticos de los modelos de iluminación del *rendering*, son más complejos que en descomposiciones regulares [TL88]. Debido a las ventajas presentadas anteriormente, el tetraedro fue escolhido como la unidad de representación en las implementaciones realizadas en este trabajo. Como en otros sistemas de *rendering* volumétrico para mallas no estructuradas [NSG90, JC94, KS00], la técnica

presentada aquí utiliza varias aproximaciones con el propósito de simplificar el proceso de visualización. La estrategia de visualización propuesta es dividida básicamente en tres etapas, que son resumidas en el diagrama del *pipeline* de visualización (*figura 1*).

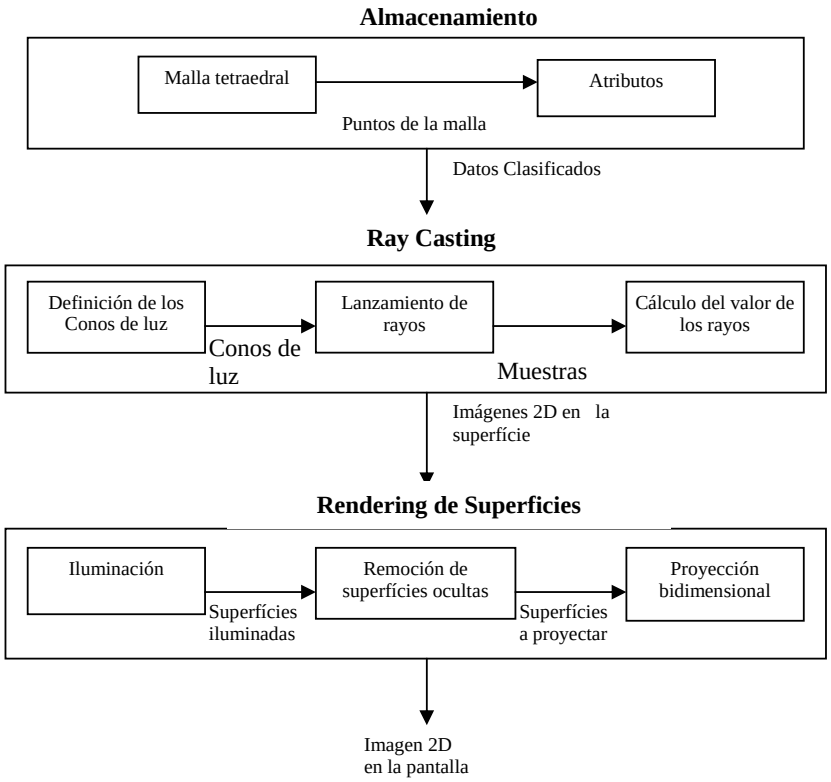


Figura 1. *Pipeline* de Visualización Híbrido**.**

Primero, los datos volumétricos son almacenados utilizando una estructura de datos topológica que representa las informaciones de incidencia y adyacencia. Luego se procede a la etapa de aplicación de la técnica de *rendering* volumétrico directo. Esa técnica es una versión modificada del algoritmo clásico de *Ray Casting* presentado en los trabajos de Levoy [Le88, Le90a]. La idea general del *Ray Casting es* tomar como plano de pre-proyección cada una de las caras del borde del volumen almacenado. A partir de cada cara es lanzado un conjunto de rayos – cada rayo con una dirección diferente – a través del volumen. Para cada rayo son calculadas las intersecciones con las células tetraedrales y tomadas muestras de color y opacidades para componer un valor escalar que será asociado con el rayo. El rayo atraviesa la malla tetraedral hasta llegar a otra cara del borde o la opacidad alcanzar un umbral determinado por el usuario.

La última etapa consiste en aplicar un método de RS. En esa etapa solamente son considerados los elementos del borde del volumen donde la pre-proyección fue realizada. A partir de la posición del observador, se utiliza uno de los escalares asociados con los rayos lanzados de las caras del borde para definir el color de la cara. De esa forma, para cada cara se adopta el color asociado a uno de los rayos de luz. Será elegido el rayo con dirección más próxima a la dirección del observador. Se puede suavizar la imagen utilizando un modelo de iluminación local sobre la superficie. La utilización del RS para la proyección final permite al usuario navegar por la escena volumétrica en tiempo real y principalmente hacer uso de *hardware* y *software* comunes, como las placas gráficas y *OpenGL* respectivamente.

A seguir describiremos más detalladamente cada una de las tres etapas del *rendering* híbrido descrito por el *pipeline.*

4.1. Almacenamiento de los Datos Volumétricos

El *rendering* de mallas irregulares ha sido identificado como una área de investigación especialmente importante en la visualización [Ka98]. En general las técnicas de *rendering* volumétrico para mallas no estructuradas utilizan una ordenación de células para calcular el orden de proyección de los elementos en el plano de visualización [SM97].

En este trabajo se considerada que la malla de entrada es almacenada en una estructura de datos topológica conteniendo las informaciones de incidencia y adyacencia para cada elemento de la malla. De esta forma, es posible recorrer las caras y elementos vecinos de cada célula de forma inmediata. Se utilizan como entrada archivos que contienen información de incidencia entre elementos (como los archivos del tipo *Unstructured Grid del VTK*). En el caso donde esa relación no es dada, es necesario realizar un pre-procesamiento para organizar los datos.

La estructura utilizada en la implementación del *rendering* volumétrico híbrido es bastante simple y representa básicamente las relaciones de vecindad necesarias para una implementación eficiente del *Ray Casting*. Particularmente, se almacena con cada vértice *v*, información de su estrella - lista de las células de la malla que contén *v* (*v* es un vértice de la célula). Se almacena también información que identifica las caras del borde del volumen (importante para las etapas posteriores del algoritmo) e indica las relaciones de incidencia entre las caras de tetraedros vecinos.

El espacio de la imagen, que normalmente consiste de N x N pixels, para el caso del proceso de *rendering* directo será substituido por las N caras del borde de los objetos de la escena.

Además de ser considerados los datos de la malla, la estructura también almacena los valores de los sus atributos. Los atributos característicos son color y opacidad. El usuario puede relacionar directamente estos atributos con cada dato del volumen, o puede agruparlos bajo una misma denominación como, por ejemplo, materiales, lo que puede caracterizar determinadas regiones. En esta etapa pueden ser utilizadas diferentes funciones de transferencia, de forma similar a la clasificación tradicional de las técnicas de visualización.

4.2. Ray Casting

Como mencionado, la base del *rendering* híbrido desarrollado en este trabajo es el *Ray Casting*, no sólo por ser una de las técnicas de visualización volumétrica directa más difundidas y conocidas, lo que facilita su entendimiento, mas también por las características y ventajas que presentan sobre otros algoritmos de visualización en mallas no estructuradas.

El *Ray Casting* fue originalmente propuesto como una técnica que permitía la visualización de pequeños detalles internos del volumen, por medio del control de la transparencia de las células.

El algoritmo de *Ray Casting* permite la generación de imágenes de alta cualidad a través del *rendering* directo, operando en el espacio de la imagen. El algoritmo clásico de *Ray Casting* se basa en rayos lanzados a partir del punto de vista del observador, que pasan a través de cada pixel de la pantalla (plano de visualización o plano de proyección) e interceptan el volumen. Para cada rayo lanzado, es tomada una muestra con base en los valores de las células y el color final de cada pixel de la imagen es obtenida integrando las contribuciones de color y opacidad de cada *voxel* interceptado por el rayo. Cuando la proyección es paralela, la dirección de estos rayos es la de la normal al plano de proyección (*figura 2*).

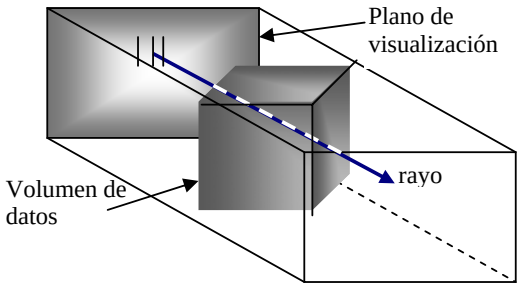


Figura 2. Algoritmo de Ray Casting

La variación del *Ray Casting* utilizada, se basa en el lanzamiento de rayos a partir de un punto en cada cara de la superficie del borde del volumen, atravesando el volumen. De esa forma, cada cara del borde es considerada como un plano de proyección, no llevando en consideración los pixels del plano de visualización. Para cada rayo lanzado, un muestreo es hecho con base en los valores de las células. El color final asociado a cada rayo es obtenido integrando las contribuciones de color y opacidad de cada célula interceptada por el rayo.

Previamente al lanzamiento de los rayos, se debe definir los "conos de rayos" – también denominados "conos de luz" – que organizan las direcciones de lanzamiento de los rayos.

Como fue observado en el *pipeline* de visualización híbrida, la etapa de Ray Casting puede ser dividida en tres módulos:

- Definición de los conos de luz;
- Lanzamiento de los rayos e intersecciones;
- Cálculo de los valores de cada rayo.

De ser necesario se puede adicionar dos módulos previos a los mencionados: un de clasificación y otro de iluminación de los datos volumétricos. La clasificación es hecha de la misma forma que para cualquier otra técnica de *rendering* que utiliza Ray Casting. Una vez que los datos fueron clasificados, se puede también aplicar un modelo de iluminación sobre ellos.

4.2.1. Definición de los conos de luz

El Ray Casting está basado en el lanzamiento de rayos a partir de un punto definido sobre cada cara del borde. Una vez definido ese punto (por ejemplo, el baricentro) es necesario determinar la dirección que ese rayo seguirá. La dirección de lanzamiento inicial es determinada por la dirección inversa a la normal de la cara, que en nuestra representación está orientada para fuera de la célula.

Una vez calculada la primera dirección ella servirá de eje para definir un nuevo sistema de coordenadas que servirá de base para el cálculo de las direcciones de los rayos que formaran el cono de luz.

O cono de luz está formado entonces por un conjunto discreto de rayos. Utilizando, por ejemplo coordenadas esféricas, pueden ser calculadas las direcciones que seguirán los rayos. Los rayos pueden tener igual latitud pero con longitudes diferentes. Una forma simple y eficiente de definir el número de rayos y de calcular sus direcciones es utilizar un padrón. El padrón sugerido es de cuatro rayos por cono, donde cada rayo sigue la dirección de uno de los ejes del sistema local de coordenadas coincidente con el plano base (fase origen) considerando las direcciones opuestas de estos ejes. Una vez definidos los rayos como vectores unitarios, se puede calcular cualquier dirección aplicando rotaciones sobre el cono, entre cero y noventa grados tanto para su longitud como latitud, intentando siempre que los rayos atraviesen la mayor parte del volumen.

Así como tenemos un conjunto discreto de rayos para un cono, podemos tener también un conjunto discreto de conos asociados a cada cara del borde. Cada cono de rayos puede tener una latitud diferente manteniendo una misma longitud, o una misma latitud y diferentes longitudes para cada rayo (*figura 3*).

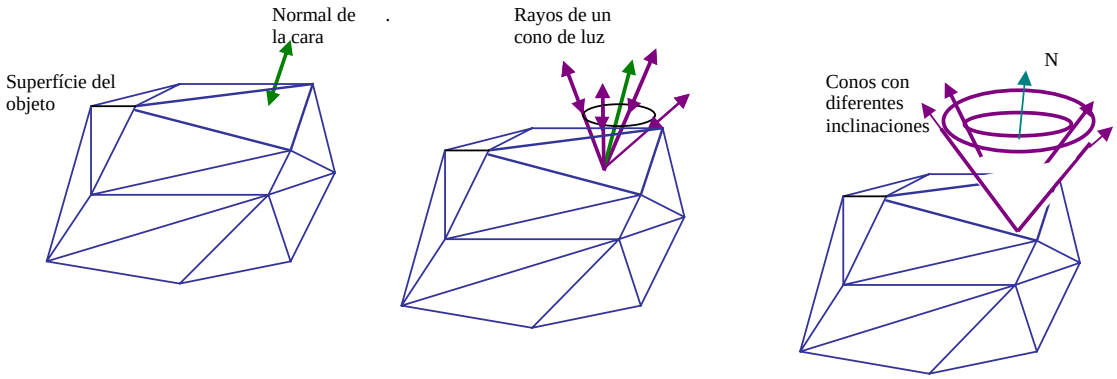


Figura 3. Definición de los Conos de Luz.

4.2.2. Lanzamiento de los rayos e intersecciones

Cada rayo es lanzado a partir de un punto conocido, luego el cálculo de las intersecciones futuras es hecho probando si el rayo atraviesa alguna de las caras de la célula en cuestión. Una vez identificado el punto, se utiliza información de la vecindad de la célula para avanzar para una otra célula en la dirección del rayo. Esto se hace hasta llegar a una otra cara del borde diferente a aquella que originó el rayo o hasta alcanzar un valor de umbral determinado por el usuario para la acumulación de la opacidad, esto es debido a que el muestreo de color y opacidad es hecho simultáneamente al cálculo de las intersecciones.

4.2.3. Cálculo de los valores de cada rayo

Una vez encontrado un punto de intersección del rayo con una cara o punto de una célula, por interpolación de los valores almacenados en los vértices de la misma, es que se calcula el valor de opacidad y color de ese punto específico y luego esos valores son compuestos junto con las otras muestras a lo largo del rayo para el cálculo del color final de la cara que origina el rayo.

El modelo considera que el volumen no atenúa la luz emitida por las fuentes de luz presentes en la escena, pues ese hecho simplifica bastante el algoritmo (no es necesario calcular una integral lineal) y evita problemas de obscurecimiento de regiones donde la densidad de los elementos es grande. Luego, el hecho de la atenuación de la luz a lo largo de los rayos lanzados no puede ser despreciado, visto que es llevado en consideración durante la etapa del cálculo del color.

Como en otras técnicas basadas en lanzamiento de rayos, la relación topológica entre las células es fundamental para un cálculo eficiente del color y opacidad a lo largo de los rayos. En el caso de las mallas no estructuradas, un aspecto importante es solucionar el problema de la complejidad de las intersecciones de los rayos con las células [BCL99], en los casos especiales, donde los rayos son paralelos a una cara, o pasan por una arista o vértice. Cabe resaltar que la restricción a células tetraedrales simplifica los cálculos en esos casos.

4.3. *Rendering* de Superficies

En esta etapa final son hechos los cálculos del *rendering* de superficies para la escena volumétrica, que permite una visualización bastante rápida. Para acelerar aun más este proceso, el *rendering* de las primitivas geométricas (fases) puede ser hecho directamente por el *hardware* disponible en las computadoras con el uso de placas gráficas comunes.

El color de cada cara es determinado de la siguiente forma: se encuentra el vector del cono de luz, de la cara, que forme el menor ángulo con la posición del observador, y se toma el color asociado a ese vector como el color de la cara.

Una vez que fueron definidos los colores de cada cara, la escena está lista para ser proyectada. Para obtener imágenes más suaves se puede aplicar un modelo de iluminación sobre las superficies de los objetos en la escena. Típicamente se utiliza el modelo de iluminación de Phong [PT75].

5. IMPLEMENTACIÓN

En la implementación de la técnica descrita en este trabajo, fue utilizado el lenguaje C++ y la biblioteca gráfica *OpenGL*. La implementación utiliza varias técnicas que toman ventaja de la coherencia espacial para reducir el tiempo de *rendering*. Así, se utiliza una estructura de datos para almacenar el modelo y aumentar la velocidad del muestreo a lo largo del rayo.

Como mencionado, para el almacenamiento de los datos, se utiliza una estructura de datos topológica que también fue implementada en C++. Una de las características de esta estructura de datos es la existencia de una “estrella del vértice”, para cada vértice *v*, que es una lista de las células que comparten *v*. Además de la estructura “vértice” se considera la estructura “fase” *f*, cuja característica principal es permitir avanzar para otra célula vecina almacenando la otra célula que comparte esta misma cara mediante un puntero.

El volumen almacenado tiene una lista de punteros a sus superficies (célula y cara del borde). Esto es útil en la hora del cálculo de la pre-imagen, en el cual serán determinados los valores de color y opacidad de la pre-imagen, para cada cara.

Para el cálculo del *Ray Casting* es utilizada el muestreo directo. Eso significa que, a la hora de calcular el punto de intersección entre las caras de las células y el rayo es hecho el muestreo de color y opacidad para el cálculo discreto del color final del rayo.

Se utiliza un umbral de opacidad para conseguir que el rayo pare en forma adaptativa. Una finalización rápida de un rayo reduce el número de células que deben ser muestreadas, y el número de comparaciones para encontrar las intersecciones del rayo con las células.

El lanzamiento de rayos se hace a partir de las caras del borde de los objetos, y no de cada pixel, siendo utilizado un número pre-determinado de rayos por cada cara, para diferentes direcciones o puntos de vista. La selección de los ángulos, que determinaran las direcciones de los rayos, así como el número de rayos lanzados a partir de cada cara, es un factor importante para la obtención de buenas imágenes, pues se debe procurar mantener un buen equilibrio entre la calidad de la imagen que se quiere obtener y el costo de memoria que requieren, tanto el cálculo de un determinado número de rayos por cara, cuanto de su almacenamiento para posteriores transformaciones.

Durante la etapa de *rendering* de superficies, dado el hecho de que las caras de las superficies a ser visualizadas son conocidas, pues fueron identificadas en etapas anteriores, sólo es necesario mantener almacenada la información de estas caras.

Particularmente, durante la etapa final de visualización, se utilizo la biblioteca *OpenGL* para proyectar las caras del borde vía *hardware*.

Cada vez que el observador cambia su punto de vista, transformaciones son aplicadas en los datos. Estas transformaciones son aplicadas, tanto sobre los puntos o vértices, como sobre las direcciones que serán comparadas para encontrar aquella más próxima al nuevo punto y/o dirección de vista del observador.

Este ambiente de visualización volumétrico híbrido facilita las transformaciones sobre las superficies de los objetos, y aprovechar mejor los diferentes recursos gráficos de la computadora, lo que nos lleva a conseguir una visualización en tiempo real después el pre-procesamiento hecho por el *Ray Casting*. Esto permite al observador navegar por la escena e interactuar con el modelo.

La *figura 4* muestra una serie de imágenes de una castaña renderizada en un ambiente de teste simple. El volumen presentado contiene 54481 células tetraedrales. La secuencia de imágenes muestra diferentes puntos de visualización. El cálculo de los colores de las pre-imágenes llevó aproximadamente 60 segundos, y la visualización de cada imagen para diferentes puntos de visualización fue en tiempo real. Como se puede observar en las imágenes, parece que estuvieramos visualizando el contenido del fruto, la semilla, pero en realidad sólo estamos visualizando la superficie del objeto.

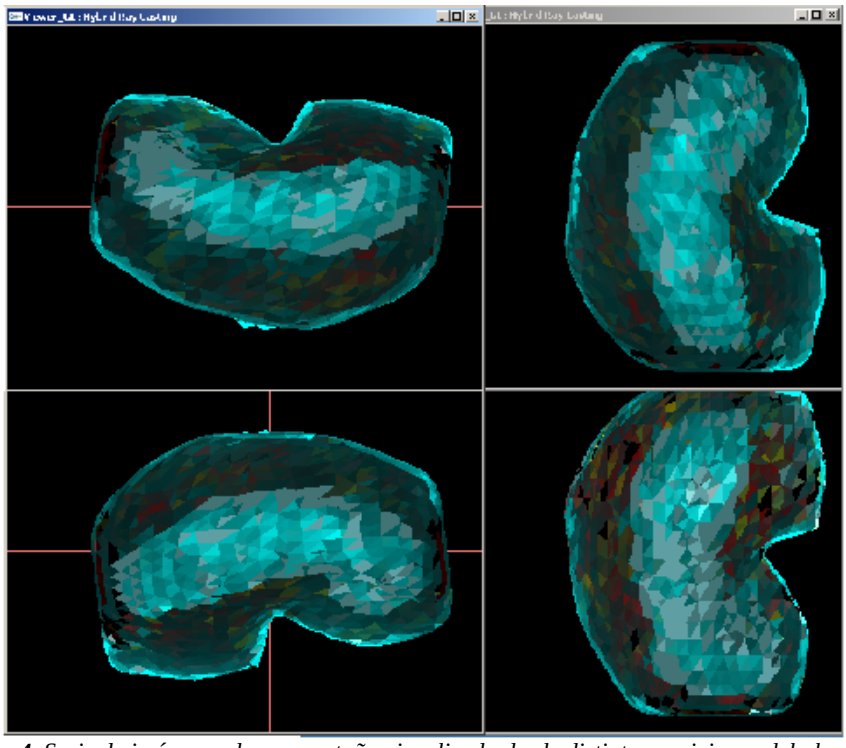


Figura 4. *Serie de imágenes de una castaña visualizada desde distintas posiciones del observador.*

6. CONCLUSIONES

La técnica de *rendering* híbrida propuesta en este trabajo aprovecha las ventajas del *Rendering* Volumétrico Directo (RVD) para la visualización de los datos internos de los volúmenes, utilizando para esto transparéncia. También son aprovechadas las ventajas del *Rendering* de Superficies (RS) para la visualización final, exhibiendo sólo las caras superficiales, las cuales almacenan la información volumétrica.

El algoritmo puede ser fácilmente paralelisado al nivel del RVD, pues el algoritmo se basa en una técnica de lanzamiento de rayos, proceso que puede ser hecho independientemente, dado que los rayos son totalmente independientes entre ellos.

El RS está basado en el *rendering* de las primitivas geométricas que forman la superficie del volumen, el que puede ser hecho directamente en *hardware* común.

Como toda técnica de visualización, esta técnica presenta algunas desventajas, tales como, que es sensible a las variaciones en los métodos utilizados durante las diferentes etapas de procesamiento. Mismo alteraciones leves pueden modificar las características observadas y la calidad de la imagen final.

Dependiendo del tipo de aplicación, y del realismo que se desee, variará directamente el nivel de refinamiento de la malla utilizada, y el número de rayos a ser lanzados a través del volumen desde cada cara del borde.

Considerando los rayos lanzados, se debe tener especial cuidado en la implementación de los algoritmos de intersección con las células, pues dependiendo de como se resuelvan los casos especiales de intersección (de las caras, aristas o vértices), artefactos pueden aparecer en la imagen final. Esto es importante, una vez que las intersecciones determinan el muestreo de color y opacidad, lo que afecta el resultado de la integración de los colores que serán dados a las caras del borde.

Utilizar conjuntos de datos con mallas refinadas permite una mejor representación de las características internas del volumen y atenúan el efecto de *aliasing* en las superficies. Sin embargo, cuanto mayor el nivel de refinamiento de las mallas, mayor es la cantidad de memoria consumida y mayor el trabajo de procesamiento necesario.

AGRADECIMIENTOS

Agradecemos al laboratorio de investigación LCAD (Laboratorio de Computación de Alto Desempeño) por su apoyo académico y de infraestructura, y al CNPq (proc. #307268/2003-9) por su apoyo financiero.

REFERENCIAS BIBLIOGRÁFICAS

[BCL99] C. L. Bajaj, E. J. Coyle and K. Lin. “Tetrahedral meshes from planar cross-sections”. Comput. Methods Appl. Mech. Eng., vol.179, pp.31-52, 1999.

[Ba00] M. Baker, “3D Theory: Spatial Decomposition”.

[CTT00] Chuan-kai Yang, Tulika Mitra, Tzi-cker Chiueh, “On-the-Fly Rendering of Loslesly Compressed Irregular Volume Data”, in IEEE Visualization '2000, Salt Lake City, Utah, October 2000.

[CMS97] P. Cignoni, C. Montani, and R. Scopigno, “Tetrahedral Based Volume Visualization,” Instituto CNUCE –C.N.R., Pisa, Italy. 1997.

[CLLC88] H. E. Cline, W. E. Lorensen, S. Ludke, D. R. Crawford, and B. C. Teeter, “Two Algorithms for Three-dimensional Reconstruction of Tomographies,” Medical Physics, vol.15, no.3, pp.320-327, Jun. 1988.

[El92] T. Elvins, “A Survey of Algorithms for Volume Visualization”, ACN SIGGRAPH, 3.-3.14 (course notes 1), 1992.

[Ga90] M. P. Garrity, “Raytracing Irregular Volume Data,” Computer Graphics (San Diego Workshop on volumen Visualization), vol. 24, pp. 35–40, Nov. 1990.

[GLDH00] M. H. Gros, L. Lippert, R. Dittrich, and S. Häring, “Two Methods for Wavelet-Based Volume Rendering”,Institute for Information Systems Computer Graphics Research Group, Swiss Federal Institute of Technology Zurich, Internal Report # 247, 2000.

[HM90] D. Helmbold and C. MacDowell, “Modeling speedup(n) greater than n”, IEEE Transactions on Parallel and Distributed Systems, vol.1, no.2, pp.250-256, April,1990.

[HL79] G. T. Herman, and H. K. Liu, “Three-dimensional display of Human Organs from Computed Tomographies,” Computer Graphics and Image Processing, vol.9, no.1, pp.1-21, Jan. 1979.

[Ho93] R. M. Hord, “Parallel supercomputing in MIMD architectures”, CRC pres, Inc.,1993.

[JC94] M. W. Jones and M. Chen. “A New Approach to the Construction of Surfaces from Contour Data”. Proc. EUROGRAPHICS94, vol.13, no.3, pp. 75-84, 1994.

[Ka91] A. Kaufman, “Introduction to Volume Visualization,” Volume Visualization, A.Kaufman (ed.), IEEE Computer Society Pres, pp. 1-18, 1991.

[Ka98] A Kaufman, “Advances in Volume Visualization”, SIGGRAPH’98, Course Notes no.24, Orlando, 1998.

[KS86] A. Kaufman, and E. Simony “Scan-conversion Algorithm for Voxel-base Graphics,” Proceeding, ACN Workshop on Interactive 3D Graphics, Chapel Hill, NC, pp. 45-75, Oct. 1986.

[Ke95] E. Keepler, “Approximating Complex Surfaces by Triangulation of Contour Lines,” IBN Journal of Research and Development, vol.19, no.1, pp.2-11, Jan. 1995.

[KS00] R. Klein, A. Schilling, and W. Straber. “Reconstruction and Simplification of Surfaces from Contours”. Graphical Models, vol.62, pp.429-443, 2000.

[LL94] P. Lacroute and M. Levoy , “Fast Volume Rendering using la Shear-Warp Factorization of the Viewing Transformation”, Computer Graphics, vol.28, no.3, pp.451-458, Jul. 1994.

[Le88] M. Levoy, “Volume Rendering: Display of Surfaces from Volume Data”, IEEE Computer Graphics and Applications, pp. 29-37, May. 1988.

[Le90a] M. Levoy, “A Hybrid Ray Tracer for Rendering Polygon and Volume Data”, IEEE Computer Graphics and Applications, vol.10, no.3, pp. 33-40, Mar 1990.

[Le90b] M. Levoy, “Ray Tracing of Volume Data”, SIGGRAPH’90 course notes, Volume Visualization Algorithms and Architectures, August,1990,pp.120-147.

[LC87] W. E. Lorensen and H.E. Cline, “Marching Cubes: la High Resolution 3D Surface Construction Algorithm, Computer Graphics (Proc. SIGGRAPH), pp.163-169, Jul. 1987.

[NSG90] K.L. Novins, F. Sillion, and D.P. Greenberg, “An Efficient method for Volume rendering using perspective projection”, Computer Graphics, vol.24,no.5,pp 95-102, 1990.

[NMS00] L.G Nonato, R. Minghim, and M.H. Shimabukuro. “Qualitative Analysis of two Reconstruction Techniques and their application in Dentistry”. Journal of Electronic Imaging, pp. 385-393, 2000.

[OUT85] T. Ohashi, T. Uchiki, and M. Tokoro, “A three-dimensional shaded display method for voxel-based representations”, in Proceedings of EUROGRAPHICS 1985, pp.221-232, National computer graphics association, September, 1985.

[PT75] Phong, Bui Tuong, “Illumination for Computer Generated Pictures”, Communications of the ACM, vol.18, no.6, pp.311-317, Jul. 1975.

[RW92] S. Ramamoorthy and J. Wilhelms, “An Analysis of Approaches to Ray-Tracing Curvilinear Grids”, Tech Report UCSC-CRL-92-07, U. of California, Santa Cruz, 1992.

[SFLP00] R. Samanta, T. Funkhouser, K. Li, and J. Pal Singh, “Hybrid Sort-First and Sort-Last Parallel Rendering with la Cluster of PCs”, Princeton University, 2000.

[ST90] P. Shirley and A. Tuchman, “A Polygonal Approximation to Direct Scalar Volume Rendering”, Computer Graphics, vol. 24, no.5, pp.63-70, Nov. 1990.

[SM97] C. Silva and J. Mitchell, “The Lazy Sweep Ray Casting Algorithm for Rendering Irregular Grids”, IEEE Transactions on Visualization and Computer Graphics, vol. 3, no. 2, Jun. 1997.

[TPN93] D. Tost, A. Puig, and I. Navazo, “Visualization of mixed scenes based on Volume and surfaces”, Fourth Eurographics workshop on Rendering, pp.281-293, 1993.

[TL88] T. Totsuka and M. Levoy, “Frequency Domain Volume Rendering”, Computer Graphics (Proc. SIGGRAPH), pp.59-64, 1988.

[Wi92] P. Williams, “Visibility Ordering Meshed Polyhedra”, ACN Transactions on Graphics, vol. 11, no. 2, 1992.

[YRLS96] R. Yagel, D. Reed, A. Law, P-W. Shih, and N. Shareef, “Hardware Assisted Volume Rendering of Unstructured Grids by Incremental Slicing”, IEEE-ACN Volume Visualization Symposium, pp. 55–62, Nov. 1996.

Estendendo o Modelo de Máquina Geométrica a um Ambiente de Programação Visual

Diego Galho Prestes, Marcos Borba Cardoso,
Renata Hax Sander Reiser, Antônio Carlos da Rocha Costa
Universidade Católica de Pelotas, Escola de Informática - NAPI,
Pelotas, Brasil, 402
{dprestes, mbcardo, reiser, rocha}@atlas.ucpel.tche.br

Abstract

In this paper we outline the main characteristics related to the implementation of a visual programming environment, denoted by APV-MG and designed to support the programming in the theoretical Geometric Machine Model. In the development of the environment APV-MG, the traditional techniques of textual programming language are combined with the construction of graphic/visual environments in order to improve the programming in the Geometric Machine and make easy the expression and simulation of parallel and non-deterministic computations. The environment APV-MG provides the visual representation of processes and constructors interpreted in the Geometric Machine Model, allowing the generation and construction of new ones. The visual language is based on well-defined concepts of algebraic graphic transformation, allowing (spatial and temporal) semantic specification of a process in a bi-dimensional way.

Keywords: Programming Environments, Visual Languages, Non-determinism, Python Programming Language.

Resumo

Neste artigo são descritas em linhas gerais as principais características relacionadas com a implementação do ambiente de programação visual, denominado APV-MG e projetado para dar suporte a programação no Modelo de Máquima Geométrica. No desenvolvimento do ambiente APV-MG, as técnicas tradicionais da programação textual foram combinadas com as construções visuais para incentivar a programação na Máquina Geométrica e tornar mais fácil a expressão e simulação de computações paralelas e não-determinísticas. O ambiente APV-MG provê representação visual para processos e construtores interpretados no Modelo de Máquina Geométrica, permitindo a geração e construção de novos processos. A correspondente linguagem visual está baseada nos conceitos consolidados de transformações algébricas sobre grafos, que permitem a especificação semântica de um processo de forma bi-dimensional.

Palavras chaves: Ambientes de Programação, Linguagens Visuais, Não-Determinismo, Linguagem de Programaçao Python.

1 Introdução

Baseado na especificação sintática da linguagem visual e textual induzida pela estrutura ordenada do modelo de Máquina Geométrica (modelo MG), este trabalho introduz o Ambiente de Programação Visual (APV-MG) para modelagem e simulação de algoritmos da Computação Científica. As etapas de desenvolvimento deste trabalho relacionam-se, principalmente, com a estruturação e implementação das seguintes ferramentas: (1) editor de processos, incluindo a definição dos testes e construtores de processos; (2) editor de memórias, modelado por uma estrutura matricial; (3) simulador, viabilizando a simulação das computações dos processos previamente construídos na interface gráfica do editor de processos e a partir da configuração da memória construída pela interface gráfica do editor de memórias.

Estas ferramentas estão interligadas pelas bibliotecas de manipulação das correspondentes representações visuais e podem ser acessadas pela interface gráfica. Salienta-se que tais ferramentas são construídas como módulos independentes, possibilitando, por exemplo, que o simulador possa utilizar as bibliotecas numéricas de outras linguagens (Python, Haskell, Intval, C++) reforçando a diversidade das futuras aplicações.

A ferramenta computacional APV-MG é capaz de estimular a programação com ênfase nas computações não-determinísticas e no paralelismo síncrono, usufruindo de uma sintaxe simples, alcançada pela utilização de técnicas de programação visual que facilitam o entendimento e incentivam sua aplicação na modelagem de sistemas.

Esse trabalho é desenvolvido na concepção de software livre e os diversos módulos da ferramenta APV-MG são implementados na linguagem Python. Tomaram-se como base resultados anteriormente alcançados e relacionados com a especificação sintática da Linguagem Visual para o Modelo Máquina Geométrica (LVMG) [9, 17]. A linguagem de programação Python, caracterizada como uma linguagem de alta portabilidade, além de apresentar facilidades no desenvolvimento de algoritmos possui várias bibliotecas gráficas disponíveis na web.

As seções deste trabalho mostram a evolução do Projeto a partir da construção do modelo MG, resumido na Seção 2. A proposta de memória, processos e testes espacialmente distribuídos conduz ao estudo de técnicas de programação visual para especificação da linguagem LVMG. Na Seção 3, apresenta-se uma extensão da especificação desta linguagem com ênfase no não-determinismo. Seguem-se a descrição da ferramenta APV-MG, os aspectos mais relevantes de sua implementação e observações sobre sua funcionalidade. As conclusões apresentam os resultados parciais e as aplicações em desenvolvimento.

2 O Modelo de Máquina Geométrica

Nesta seção serão resumidos alguns conceitos básicos do modelo MG necessários para a descrição da metodologia de desenvolvido da ferramenta APV-MG.

Os Espaços Coerentes [4, 5, 6] introduzidos primeiramente para prover uma semântica denotacional para a Lógica Linear, foram utilizados para caracterizar a estrutura ordenada do modelo MG [11], com o objetivo de obter uma representação para computações concorrentes e (ou) não-determinísticas. Neste modelo, as representações da memória e dos processos computacionais, possivelmente infinitos, envolvendo paralelismo e(ou) não-determinismo, estão rotulados por posições de um espaço geométrico.

A partir desta distribuição espacial, obteve-se uma integração de conceitos importantes da Teoria da Concorrência e da Álgebra de Processos [8], pois a noção de computação neste modelo é concebida como uma transição de estados associada a uma localização espacial, capaz de modelar de forma explícita a concorrência síncrona e o conflito de acesso à memória.

Por outro lado, induzida pela estrutura ordenada e fundamentada na Teoria dos Domínios [19], foi definida uma linguagem de programação textual possibilitando o desenvolvimento de programas para o modelo MG[14] e sua correspondente semântica denotacional. A linguagem suporta a representação de processos parciais, modelando os estados parciais da computação. Pelo procedimento de completação são preservados os limites de computações infinitas, garantindo que a solução para equações recursivas, definidas sobre os construtores do modelo MG, sejam expressões válidas na linguagem textual.

Entretanto, no modelo também podem ser representados programas com estrutura lógica mais complexa, compatível com as novas arquiteturas e sistemas de organização de computadores, como o processamento paralelo síncrono e(ou) distribuído [12]. Neste caso, o modelo MG salienta-se pela uniformidade da estrutura ordenada que possibilita tais representações, significando uma forte e direta relação entre estes sistemas geralmente tratados de forma isolada na literatura.

A natureza indutiva da estrutura ordenada do modelo MG permitem a modelagem de construções recursivas, aplicadas às computações (não) determinísticas, incluindo dois tipos especiais de paralelismo: o espacial, com memória e processos possivelmente infinitos, definidos por estruturas matriciais; e o temporal, na versão genérica do modelo, com memória global transfinita e processos distribuídos num conjunto enumerável de modelos MG, sincronizados no tempo [3].

Os estudos e as aplicações na modelagem de algoritmos para Matemática Intervalar, Processos Estocásticos, Computação Quântica e Autômatos Celulares [10, 13, 15, 16], mostram que o modelo MG é capaz de prover uma análise semântica para algoritmos (recursivos) da Computaçao Científica.

3 Linguagem Visual para a Ferramenta APV-MG

A especificação da LVMG, cuja sintaxe segue o modelo proposto no Projeto GENGED [1], inclui a definição da sintaxe abstrata (responsável pela estrutura lógica das expressões gráficas) e da sintaxe concreta (orientando o layout de cada expressão para manipulação do usuário). Da mesma forma é construída a especificação da gramática visual, integrando conceitos da Teoria dos Grafos [20, 21] na representação dos construtores: produto seqüencial e paralelo, somas determinística e não-determinísticas, e as correspondentes construções recursivas. As regras que compõe a gramática visual são definidas em duas etapas distintas: a preparação (caracterizada pela representação de objetos parciais) e a realização.

Nesta abordagem, o alfabeto visual é representado por um grafo, e a gramática visual é representado por uma gramática de grafos. O alfabeto visual estabelece o conjunto de símbolos e conexões utilizados na especificação de uma linguagem visual, ou seja, define o vocabulário da ferramenta LVMG. A construção de cada símbolo do alfabeto consiste na construção de um grafo onde nodos e arestas estão relacionados por atributos e conexões. A correspondente gramática visual é obtida a partir de um diagrama inicial e um conjunto finito de regras que geram outros diagramas da ferramenta LVMG, que no caso representam os processos no modelo MG.

3.1 Especificação da Sintaxe para a Ferramenta APV-MG

Em [9, 17] introduz-se a especificação dos construtores da linguagem responsáveis pela sequencialidade e sincronização de processos (produto sequencial e produto paralelo) incluindo suas construções recursivas [3]. Estendendo esta especificação, este trabalho apresenta os construtores soma determística e soma não-determinística, cuja especificação também está dividida em duas atividades principais: (1) Especificação do Alfabeto, composto pela Especificação de Símbolos e pela Especificação de Conexões; (2) Especificação de Gramáticas, onde são construídas as regras que definem os demais construtores de processos do modelo MG: soma determinística e soma não-determinística.

3.2 Exemplificando a Especificação do Alfabeto para a LVMG

Neste trabalho, aplica-se aos processos do tipo soma não-determinística e soma determinística a mesma metodologia de construção da sintaxe apresentada em [9].

3.2.1 Processo Soma Não-Determinística

Na Figura 1(a), tem-se a representação gráfica de uma escolha não-determinística. Seu grafo possui dois nodos ou objetos (processos representado por retângulos na sintaxe concreta) que correspondem aos termos, juntamente com dois arcos (conexões), ambos com origem no nodo-processo e denominados "termo_sup" e "termo_inf". Na sintaxe concreta, o grafo do processo não-determinístico é formado pela inclusão dos subgrafos de cada termo a partir da função de inclusão, indicada por arcos tracejados e denominada "insere_nd". Além disso, a expressão textual de cada um dos termos é mapeado para a sua localização na expressão gráfica do processo não-determinístico pela aplicação da função "incluind_nome", considerando condições quanto ao tamanho e ao nome da fonte utilizada.

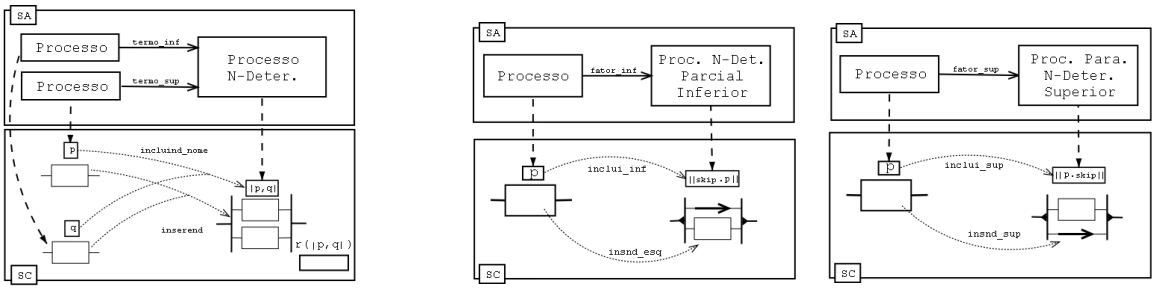


Figura 1: Grafos de Representação para Soma Não-Determinística

Observa-se que no canto inferior direito está representada a função posição, verificando o conflito de acesso à mémoria. Neste caso, tem-se processos distintos com mesma indexação.

Nas Figuras 1(b) e 1(c), tem-se a especificação da sintaxe visual e concreta dos correspondentes processos parciais associados a soma não-determinística, que serão posteriormente aplicados na fase de preparação da regra gramatical para o construtor soma não-determinística, conforme mostram os exemplos na Seção 3.3.1.

3.2.2 Processo Soma Determinística

Pela abordagem não-determnística do modelo MG, as escolhas determinísticas podem ser definidas por testes determinísticos ou não. No segundo caso, além dos testes existenciais, que ocorrem quando pelo menos um dos processos de entrada satisfaz a propriedade que define o teste, tem-se também os testes universais, obtidos na concepção dual. A especificação sintática do objeto processo soma determinística, resultante da

aplicação do construtor soma determinística é descrita nos próximos parágrafos, e uma instanciação está representada na Figura 2.

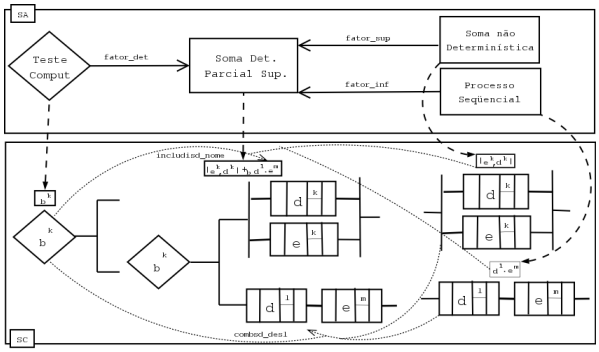


Figura 2: Grafo de Representação para Soma Determinística

Pela sintaxe abstrata, o diagrama de representação da soma determinística é definido sempre que for possível explicitar três objetos: dois processos e um teste (universal, existencial ou determinístico) juntamente com os conectores que o constroem. Tanto o primeiro quanto o segundo fator da soma determinística são representados por objetos (processos) acrescidos dos conectores "fator_sub" e "fator_inf", respectivamente. Estes objetos consistem em subgrafos gerados pela linguagem visual, ou seja, são elementos do alfabeto visual representando processos parciais. Na sintaxe concreta, a soma determinística é representada por dois nodos-processo que se ligam a um nodo-teste formando um novo processo. O diagrama é composto também pelas funções de inclusão, de dois argumentos, identificadas neste caso pelas expressões "incluisd_nome" e "combsd_desl". Próximo a cada nodo do grafo soma determinística fica indicada a expressão associada ao processo na linguagem textual. De forma análoga, cada tipo de teste será representado por um objeto (teste).

Os processos acima apresentados são parte do abrangente vocabulário associado à Linguagem Visual definida para a Máquina Geométrica, incluindo outros construtores e suas expressões recursivas e parciais.

3.3 Exemplificando a Gramática para a Linguagem Visual

Neste trabalho, as primeiras regras de construção de objetos referem-se a soma não-determinística e estão divididas em duas etapas, identificadas como a preparação e a realização. Cada etapa é modelada por uma transformação entre subgrafos definidos no vocabulário da LVMG. Salienta-se, entretanto, que para todos os construtores representados no modelo MG tem-se a correspondente regra na gramática da LVMG.

3.3.1 A Regra de Construção da Soma Não-Determinística

Na transformação de grafos que formaliza a etapa de preparação, a sintaxe abstrata dos processos são representados por subgrafos identificados pelos arcos de conexão: "termo_sup" e "termo_inf". Na sintaxe concreta a soma não-determinística é representada por diagramas entre barras verticais paralelas onde são inseridos os objetos (processos) em sobreposição vertical. Estas sobreposições são identificados pelas funções: "nomend_sup", "nomend_inf" e "insnd_sup", "insnd_inf". Na Figura 3, mostram-se os objetos e as conexões envolvidos na transformação dos grafos desta subetapa, incluindo os correspondentes morfismos envolvidos na gramática. A janela que se localiza na parte inferior, mostra que a condição para aplicação da etapa de preparação deve ser satisfeita e explicitada na tela: $\Gamma(p) \bigcap \Gamma(Skip) \neq \emptyset$ ou $\Gamma(Skip) \bigcap \Gamma(q) \neq \emptyset$ ($Skip \equiv Id$).

Na etapa de realização, estes subgrafos (processos parciais) são reunidos num único grafo pelas funções binárias que inserem objetos-processos: "incluind" e "combnd". O diagrama de representação para a gramática de um processo não-determinístico na fase de realização está representado logo a seguir, na Figura 4. A janela que se localiza no extremo esquerdo inferior mostra ao usuário quais as posições de memória afetadas na execução do processo que está sendo gerado, que devem satisfazer a condição $\Gamma(p) \bigcap \Gamma(q) \neq \emptyset$.

3.3.2 A Regra de Construção da Soma Determística

Na etapa de preparação tem-se a construção dos grafos que modelam os termos parciais da soma, referidos como soma determinística parcial superior e soma determinística parcial inferior. Cada um deles está representado por dois sub-grafos onde um dos grafos representa sempre o Processo Identidade. Também

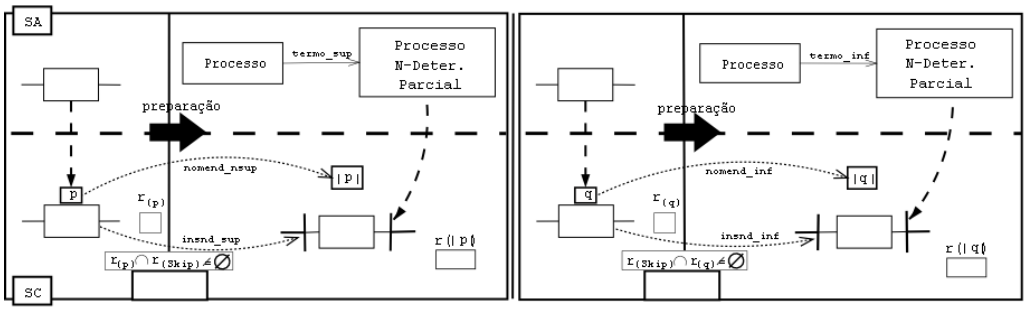


Figura 3: Etapa de Preparação da Regra Gramátical da Soma Não-Determinística

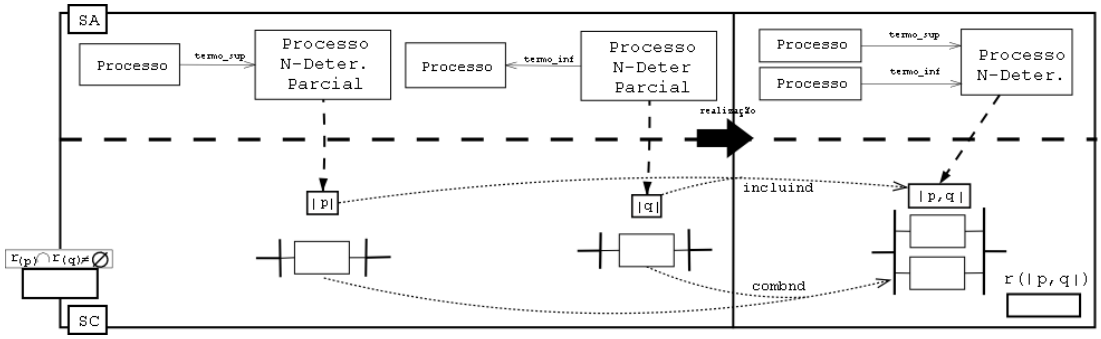


Figura 4: Etapa de Realização da Regra Gramátical da Soma Não-Determinística

são termos parciais da soma determinística os testes. Esta etapa constitui-se numa condição que deve ser previamente satisfeita para aplicação da soma determinística. Veja detalhes na Figura5.

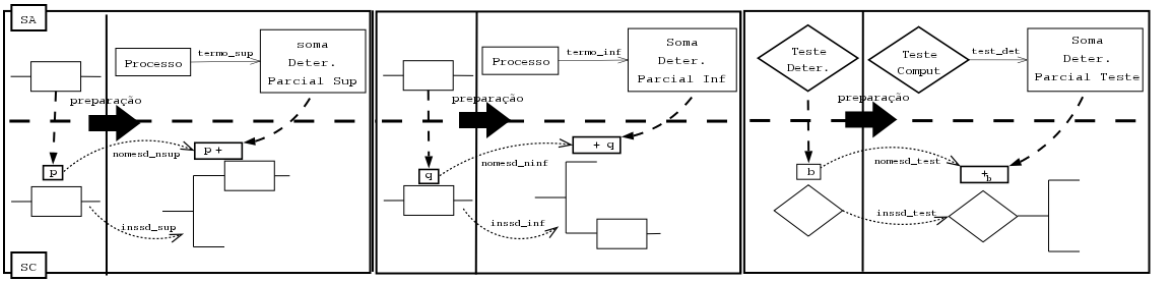


Figura 5: Etapa de Preparação da Construção da Soma Determinística

A etapa de realização está apresentada na Figura 6. As funções de dois argumentos que retornam um argumento e estão envolvidas na especificação desta etapa são: “incluisd_nome” e “combsd_desl”.

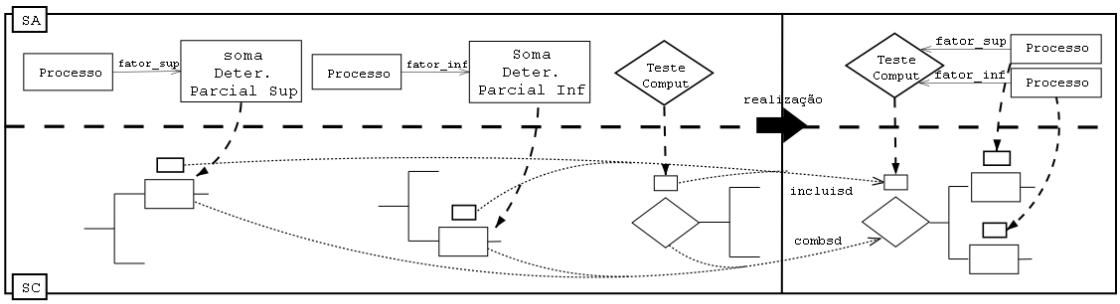


Figura 6: Etapa de Realização da Soma Determinística

4 Descrição da Ferramenta APV-MG

Desenvolvida com o objetivo de estimular a simulação de algoritmos paralelos e distribuídos da Computação Científica, esta ferramenta computacional está implementada com base na concepção livre, multi-plataforma, utilizando a linguagem Python e aplicando técnicas visuais para construção gráfica de processos e da memória.

A modelagem da memória por uma estrutura matricial representa a discretização do espaço geométrico. A indexação dos valores de memória explicitada na representação gráfica, facilita a aplicação e compreensão do paralelismo síncrono e do não-determinismo caracterizado pelo conflito de acesso à memória, além de permitir a visualização da simulação das computações.

As principais etapas da construção da ferramenta APV-MG são brevemente esquematizadas na Figura 7 e consistem, principalmente, na construção e implementação do editor de processos, do editor de memórias, do simulador e das correspondentes interfaces gráficas que viabilizam o acesso ao usuário do ambiente.

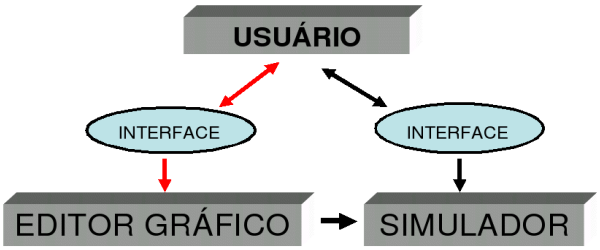


Figura 7: Diagrama do APV-MG

Através do editor gráfico pode-se efetivar a manipulação de arquivos, com opções para criar, deletar, salvar, redimensionar e reconstruir objetos gráficos. Na operacionalização destas funções utilizou-se o conceito de envelopes. Os envelopes são arquivos gerados na linguagem Python para representação interna dos processos, caracterizados pelo agrupamento dos objetos gráficos na interface do Editor de Processos. Estes arquivos são gerados e implementados no módulo de Representação Interna dos Processos.

Na Figura 8, tem-se um esquema do ambiente com especificação dos módulos e das comunicações entre o editor de processos, o editor de memórias e o simulador.

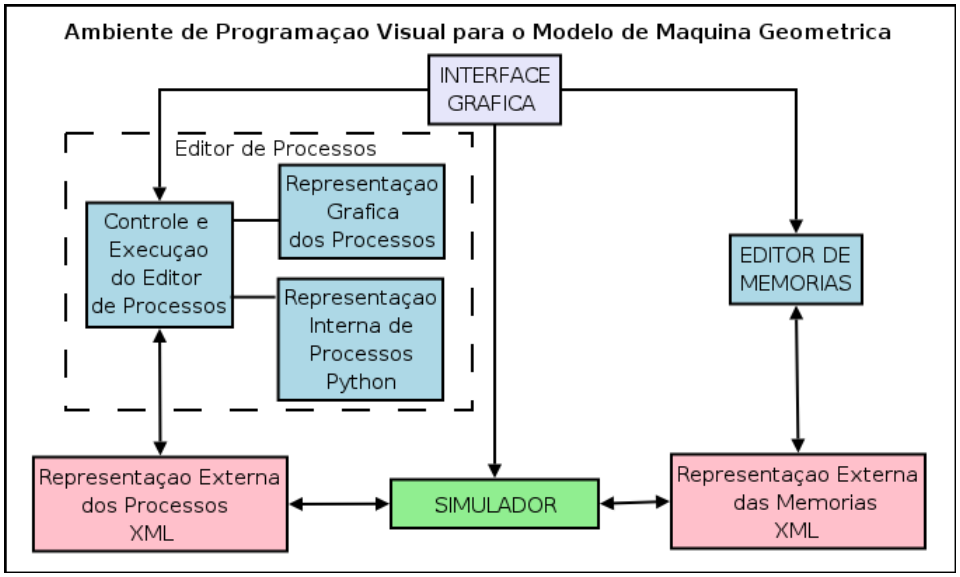


Figura 8: Esquema do Editor de Processos

Os objetos gráficos também são descritos por arquivos gerados na linguagem XML e armazenando no módulo de representação externa dos processos. A coleção destes arquivos constitui a biblioteca dos processos, cujo acesso é viabilizado pela interface do editor de processos ou pela interface do simulador.

O módulo de execução e controle do editor de processos permite a comunicação entre as diferentes representações dos objetos gráficos, ou seja, os correspondentes arquivos em XML e em Python.

A configuração dos processos enviada ao módulo de controle e execução do simulador está especificada nesta descrição em XML dos objetos gráficos.

O editor de memórias possibilita a representação gráfica das memórias, disponibilizando ao usuário, além da escolha da estrutura matricial, possivelmente multidimensional e adequada ao desenvolvimento de uma aplicação, a visualização das tranformações das memórias nas computações geradas na simulação.

Apesar de serem omitidos do esquema apresentado na Figura 8, a modularização do editor de memórias para representação interna e externa de estados, para criação de funções de controle e execução, assim como sua integração com o simulador, podem ser construídos de forma análoga ao editor de processos.

Salienta-se que, pelo módulo de execução e controle do editor de memórias são exportados os arquivos em Python para o módulo de representação externa de memórias, responsável pela geração e armazenamento das correspondetes descrições em XML, formando uma biblioteca de configurações de memória.

O processo dinâmico de simulação, apresentado no esquema da Figura 9, ocorre a partir da configuração dos processos e das memórias, definidos neste caso como parâmetros de entrada.

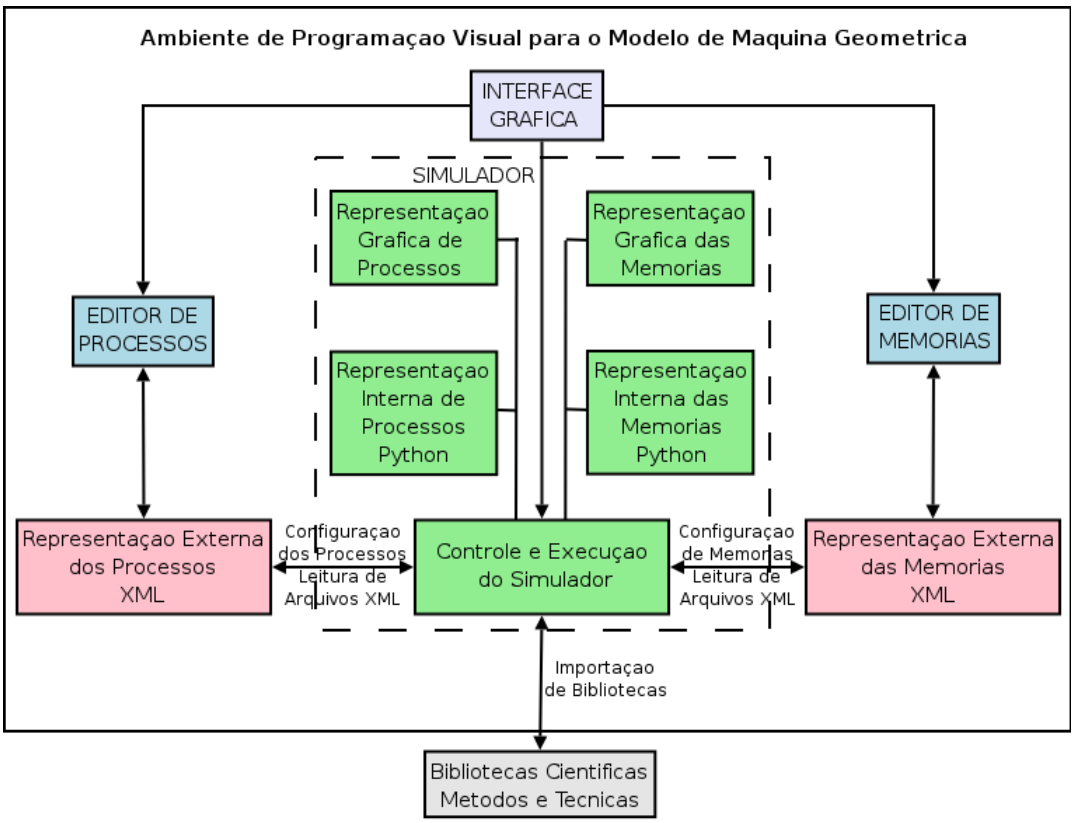


Figura 9: Esquema do Módulo Simulador

O módulo de controle e execução do simulador importa os arquivos de outras bibliotecas (numéricas) da Computação Científica provendo os recursos (métodos, técnicas, funções e procedimentos) necessários para a simulação. O ambiente permite que os resultados parciais e finais dessa simulação sejam disponibilizados para o usuário pela interface do simulador, cuja descrição e detalhes são apresentadas na Seção 6.

5 Implementação da Ferramenta APV-MG

O desenvolvimento do Ambiente de Programação para o Modelo MG consiste em diferentes etapas, dentre as quais destacam-se a implementação do alfabeto e da gramática, com a definição dos construtores.

5.1 Linguagem de Programação Python

A implementação da ferramenta está sendo desenvolvida na linguagem Python, caracterizada como uma linguagem orientada a objetos e de fácil aprendizagem, possuindo bibliotecas de diversos tipos de dados e para quase todos tipos de aplicações envolvendo Computação Científica e Comercial [2, 18].

Neste trabalho, utilizam-se as potencialidades de sua biblioteca gráfica wxPython, de fácil aprendizagem e possibilitando a criação de uma interface visual para o usuário. O estudo da OGL (Object Graphics Library) possibilitou a construção e manipulação dos grafos, incluindo a criação de objetos geométricos (como linhas, quadrados, círculos) utilizados na representação dos processos do Modelo MG. Essa biblioteca, originalmente desenvolvida em C++, possui um suporte para o wxPython e, atualmente, é uma biblioteca em código Python puro. Assim como o Python, as bibliotecas wxPython e OGL têm código-fonte aberto, multiplaforma, ou seja, a ferramenta APV-MG poderá rodar em diversas plataformas, sem precisar de modificação ou adaptação.

5.2 Linguagem de Marcação XML

A utilização do XML (Extended Markup Language) [7] possibilitou a representação dos processos do Modelo MG em forma de dados estruturados, viabilizando a comunicação entre o Editor e o Simulador. Neste sentido, o Simulador poderá receber um arquivo que representará um processo por um conjunto de programas. Este arquivo junto com a memória de entrada, é executado pelo compilador da linguagem Python, viabilizando assim a simulação do processo representado no editor da ferramenta APV-MG. É possível também utilizar esse mesmo arquivo XML dentro do próprio editor, com a chamada do processo já construída e previamente salva. Dentre as bibliotecas disponibilizadas, foi estudada primeiramente a biblioteca do SAX (Simple API for XML), mais simples e fácil de implementar. Posteriormente, o Projeto optou pela biblioteca DOM (Document Object Model), capaz de prover uma estrutura de árvores para percorrer arquivos gerados pelo XML.

O conjunto de tecnologias utilizadas no desenvolvimento dos editores e do simulador para a ferramenta APV-MG está esquematizado na Figura 10. Estas funções são referenciadas quando se faz a descrição do funcionamento do ambiente.

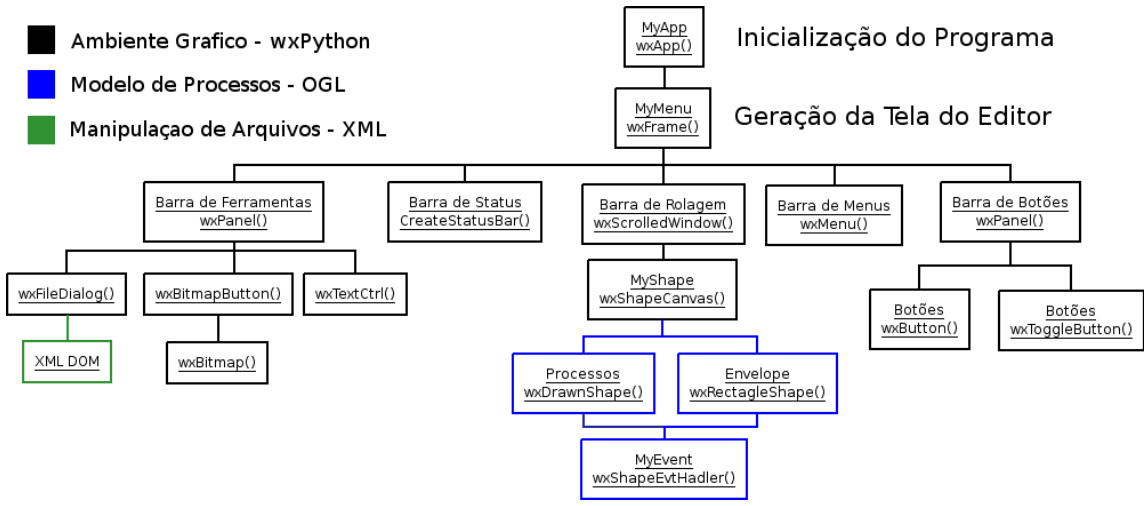


Figura 10: Tecnologias Aplicadas no Desenvolvimento da Ferramenta do APV-MG

6 Funcionalidades da Ferramenta APV-MG

Nesta seção são brevemente descritas as funcionalidades da ferramenta APV-MG.

6.1 Criação e Geração das Interfaces para Editores e Simulador

Os Editores de Memórias e de Processos viabilizam, através de suas interfaces gráficas, a configuração dos estados computacionais assim como a criação e geração de processos e testes pela aplicação dos construtores: produto paralelo, produto seqüencial, a soma determinísticas e soma não-determinística, incluindo ainda as correspondentes versões iterativas e recursivas.

A fase inicial do desenvolvimento do editor foi centrada na criação e geração da interface para Ferramenta APV-MG. Destacam-se o estudo e a aplicação das seguintes funções da biblioteca wxPython: (1) função wx.App, procedendo a chamada para a tela do programa, (2) função wx.Frame que contém as informações necessárias para a criação e geração de janelas, cujos principais parâmetros são posição na tela, tamanho e título de identificação.

A inserção de botões na tela permite o acesso a funções de criação do botão padrão (wx.Button), condicionadas pela especificação da posição da janela, do tamanho e do texto de identificação da função do botão. Utiliza-se uma função de controle juntamente com um botão especial (wx.ToggleButton) que disponiliza estados que são utilizados na seleção dos processos a serem construídos. Para geração de código e criação da tela do Editor de Processos, apresentada na Figura 11, foram usados componentes das bibliotecas wxPython e OGL, possibilitando o acesso a funções do programa através de botões e das barras de ferramentas.

Dependendo das aplicações, o Editor de Memórias disponibiliza as opções para escolha da memória, incluindo a dimensão da estrutura matricial e a especificação dos valores de memória, veja Figura 12(a). Ao lado, a Figura 12(b) mostra a interface do Simulador.

```
menu = wxMenu() #### Criação de Menus
menu.Append(ID_NOVO, "&Novo", "Comeca novo projeto")
menu.Append(ID_ABRIR, "&Abrir", "Abre um projeto")
menu.Append(ID_SALVAR, "&Salvar", "Salva projeto")
menu.AppendSeparator()
menu.Append(ID_EXIT, "Sai&r", "Finaliza o programa")
.
.
.
#### Criação da Barra de Ferramentas
self.ferr = wxPanel(self, 30, wxPoint(0,0), wxSize(300,30), wxRAISED_BORDER)
bmp1 = wxBitmap("images/novo.bmp", wxBITMAP_TYPE_BMP)
self.ferr1 = wxBitmapButton(self.ferr, 31, bmp1, wxPoint(0,0), wxSize(25,25))
bmp2 = wxBitmap("images/abrir.bmp", wxBITMAP_TYPE_BMP)
self.ferr2 = wxBitmapButton(self.ferr, 32, bmp2, wxPoint(25,0), wxSize(25,25))
.
.
### Definição dos Eventos
EVT_BUTTON(self.ferr, 31, self.Novo)
EVT_BUTTON(self.ferr, 32, self.Abrir)
.
.
#### Criação da Barra de Botões
self.size = (70,270) #### Tupla que controla o tamanho da barra de botoes ####
self.leftwin = wxPanel(self,10,wxPoint(0,30),wxSize(self.size[0],self.size[1]), wxRAISED_BORDER)
bmp5 = wxBitmap("images/pt.bmp", wxBITMAP_TYPE_BMP)
self.pt = wxBitmapButton(self.leftwin, 16, bmp5 , wxPoint(20,10), wxSize(30,30))
self.left1 = wxButton(self.leftwin, 11, "PP", wxPoint(20,50), wxSize(30,30))
self.left2 = wxToggleButton(self.leftwin, 12, "SND", wxPoint(20,90), wxSize(30,30))
self.left3 = wxToggleButton(self.leftwin, 13, "SD", wxPoint(8,130), wxSize(30,30))
self.left31 = wxButton(self.leftwin, 132, "S", wxPoint(43,128), wxSize(15,15))
self.left32 = wxButton(self.leftwin, 133, "N", wxPoint(43,148), wxSize(15,15))
self.left4 = wxButton(self.leftwin, 14, "PS", wxPoint(20,170), wxSize(30,30))
self.left5 = wxToggleButton(self.leftwin, 15, "Processo\nElementar", wxPoint(3,210), wxSize(60,50))
.
.
#### Criação da Área de Trabalho
self.rightwin = MyShape.MyShape(self,20,wxPoint(self.size[0],30),wxSize(300,270))
EVT_RIGHT_UP(self.rightwin, self.Draw) #### Funcao Desenha ####
```

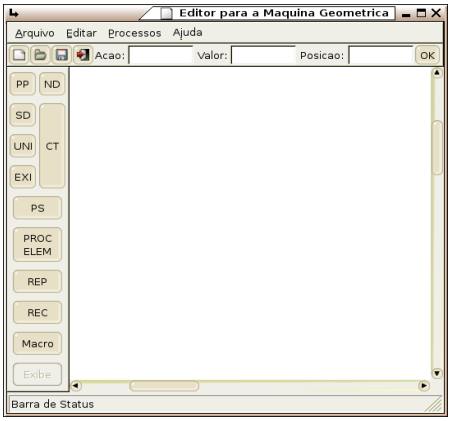
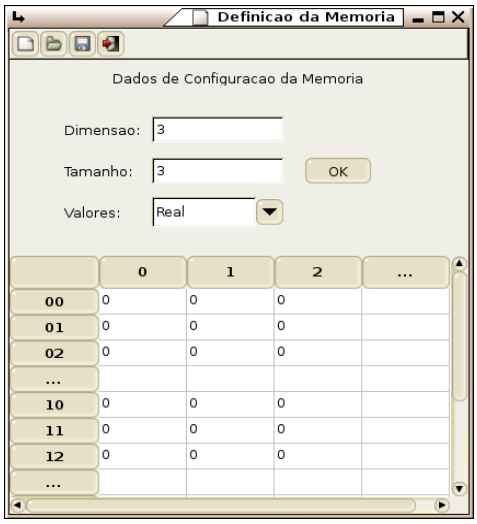


Figura 11: Visualização do Código e da Tela do Editor Processos.



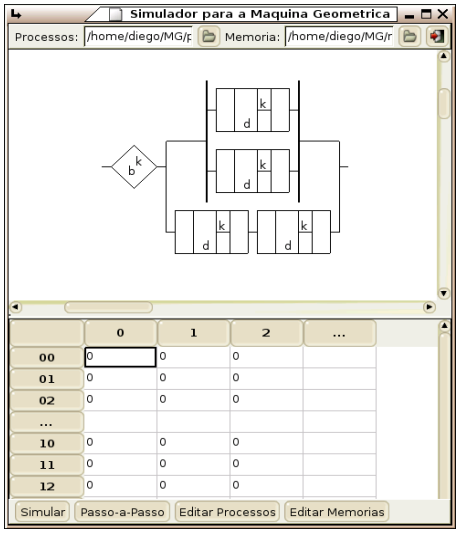


Figura 12: Interface do Editor de Memória e do Simulador

6.2 Criação e Geração de Processos e Construtores

A criação e geração de processos pela ferramenta APV-MG é bastante simples, obedecendo critérios como a modularidade, que permite ao usuário criar novos processos a partir dos processos elementares. O ambiente

possiu um módulo independente para representação gráfica de processos, os quais podem ser armazenados em uma biblioteca de processos direcionada a aplicação em desenvolvimento. Esta modularidade possibilita a disponibilidade e agiliza as buscas.

Cada processo criado no Editor pode ser salvo como arquivo XML, e a coleção de todos os processos criados pelo usário para uma determinada aplicação constitui uma coleção de arquivos XML que dão suporte à Biblioteca de Processos.

Para criação de processos basta apenas selecionar, na ordem desejada, os processos elementares existentes no editor ou já construídos na biblioteca de processos e posteriormente, escolher a opção de construtor.

Outra importante propriedade da ferramenta APV-MG é a validação dos objetos construídos no editor. Por exemplo, na Figura 13(a), pode-se visualizar a criação de uma Soma Não-Determinística, que pela sua concepção no modelo MG, só pode ser executada quando há conflito de memória entre os processos selecionados. Neste caso, a validação das condições necessárias e suficientes para execução da escolha não-determinística pelo ambiente é regulada por uma função de controle, implementando a função posição discutida na Seção 3.2.1. Esta função verifica a existência do conflito de acesso às posições de memória, modelada por uma estrutura matricial previamente definida a partir da discretização de um espaço geométrico. Quando estas condições de validação não são satisfeitas, o ambiente não permite a criação do processo e envia uma mensagem ao usuário.

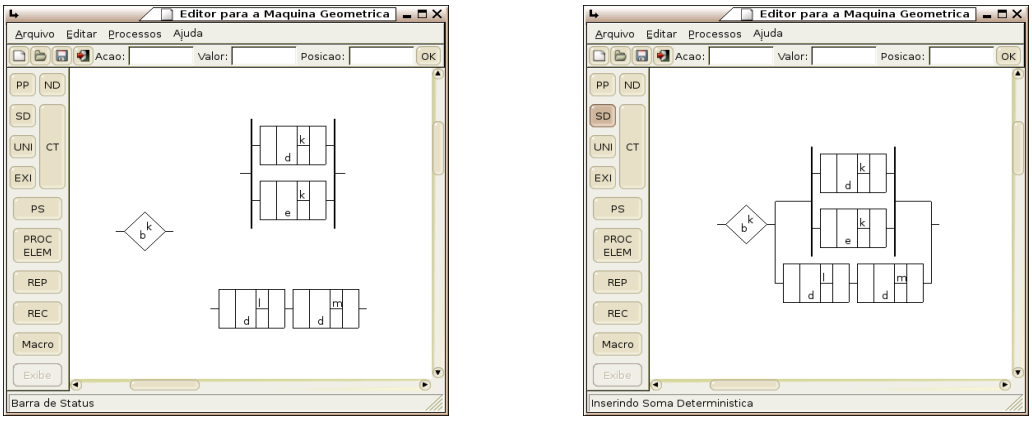


Figura 13: Visualização da Construção de uma Soma Determinística

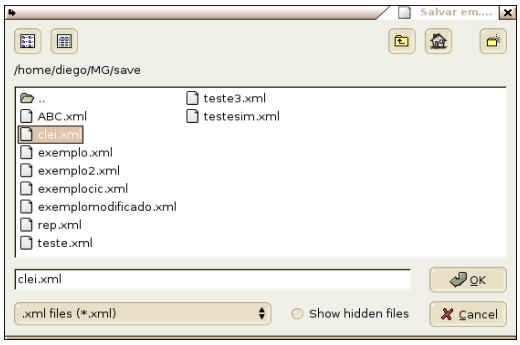
Para a criação dos objetos foi escolhida a utilização do módulo wx.DrawnShape. Nesse módulo, foram aplicados funções para criação de linhas, na qual são informados o ponto inicial e o ponto final de cada linha, ambos referenciados em coordenadas cartesianas (x,y). Desta forma, torna-se possível a criação do primeiro tipo de objeto (representação visual do processo elementar) denominado classe ConsElem. Os demais tipos de processos são gerados pelo editor de forma análoga, incluindo os processos iterativos e recursivos, os quais estão relacionados com estrutura espacial e temporal do modelo MG, e portanto definidos por estruturas mais complexas, como recorrência e macros para modelagem da recursão.

Uma parte fundamental no desenvolvimento da ferramenta APV-MG envolveu o estudo e aplicação de eventos, responsáveis pelo acionamento de funções dentro do programa. Mais especificamente, foram utilizados eventos: no acionamento e construção de botões, barras de menu e alteração no tamanho da tela, no redimensionamento dos objetos da tela e na execução de ajustes que mantêm a proporcionalidade em relação à tela.

6.3 Manipulação de Arquivos

Uma das funcionalidades adicionada ao editor foi a possibilidade de salvar e posteriormente abrir um arquivo, conforme pode ser visualizada na Figura 14. Na mesma Figura também é possível visualizar um exemplo de arquivo XML gerado pelo editor.

A biblioteca em construção para o desenvolvimento do APV-MG contém uma rotina para salvar como arquivos os objetos desenhados. O comportamento desta função inclui a leitura de propriedades e dados como posição e informações descritivas, assim como as possíveis ligações com outros objetos. Para guardar os dados foi utilizado um padrão de gravação em arquivo que permite novo acesso.




```
<save arq="/home/diego/MG/save/exemplo.xml">
 <process repr="env" tipo="seq" nome="" pos="0, 1">
  <process repr="env" tipo="paralelo"
           nome="" pos="0, 1">
   <process repr="conselem" acao="d" pos="0"
            x="1094.0" y="1119.0"></process>
   <process repr="conselem" acao="e" pos="1"
            x="1094.0" y="1189.0"></process>
  </process>
  <process repr="env" tipo="nondet"
           nome="" pos="0">
   <process repr="conselem" acao="d" pos="0"
            x="1212.0" y="1119.0"></process>
   <process repr="conselem" acao="e" pos="0"
            x="1212.0" y="1189.0"></process>
  </process>
 </process>
</save>
```

Figura 14: Tela Salvar Arquivo e Exemplo de Arquivo Gerado

7 Conclusão

O ambiente APV-MG está sendo desenvolvido com o objetivo de incentivar a prática da programação utilizando técnicas de sincronização, modelagem da memória (compartilhada) e do conflito de acesso (não-determinismo).

Alguns dos importantes benefícios do uso das representações visuais (espaciais) em modelos computacionais que envolvam programação paralela e distribuída são: o número de argumentos (aridade) envolvidos na definição dos processos algébricos, as (possíveis) definições recursivas dos construtores, a composição funcional, as computações paralelas e não-determinísticas relacionadas com a sincronização de processos, e finalmente, a distribuição espacial dos processos e estados computacionais.

Entende-se que as características da programação visual descritas acima sejam relevantes para tornar a programação no modelo MG mais acessível aos usuários em geral, possibilitando sua aplicação na Computação Científica, em especial, no desenvolvimento de algoritmos paralelos e distribuídos da Matemática Intervalar, da Computação Quântica e de Processos Estocásticos.

Na continuidade da modelagem do ambiente visual, foram apresentadas as especificações da sintaxe abstrata e concreta do alfabeto e da gramática associadas aos construtores soma determinística e soma não-determinística.

A modelagem do alfabeto viabilizou a definição e implementação do editor de processos e do editor de memórias, introduzindo a implementação do simulador.

A aplicação da biblioteca gráfica wxPython, possibilitou a manipulação de diagramas, assim como a definição e construção de funções de acesso ao ambiente, criação e redimensionamento de objetos, interação e uso de agentes e classes.

Na implementação do alfabeto e da gramática foram utilizadas estratégias da programação visual e as bibliotecas disponíveis para a linguagem Python.

Os trabalhos atuais estão centrados na conclusão e resolução de testes do módulo de controle e execução do simulador, na integração com as atividades já desenvolvidas na implementação dos editores, bem como o desenvolvimento de aplicações. Pela indenpendência dos módulos, o simulador pode ser implementado por extruturas externas mais complexa, como por exemplo em ambientes de clusterização.

Por fim, como significativa contribuição tem-se o caráter didático da ferramenta, conectando a teoria com a prática da programação. Ou seja, a partir de conceitos e fundamentos da Teoria dos Domínios obtém-se um modelo computacional, incluindo o desenvolvimento da linguagem e a implementação do ambiente computacional para simulação de processos paralelos e distribuídos.

Agradecimentos

Os autores são gratos pelo financiamento parcial da FAPERGS e do CNPq/PIBIC e CNPQ/CTINFO.

Referências

[1] R. Bardohl and C. Ermel. Visual specification and parsing of a statechart variant using genged. In *Proc. Symposium on Visual Languages and Formal Methods*, pages 5–7, 2001.

[2] M. C. Brown. *Python:The Complete Reference*. Osborne/McGraw-Hill, Berkeley, 2001.

[3] M. B. Cardoso, R. H. S. Reiser, A. C. R. Costa, and G. P. Dimuro. Especificando a recursão na linuagem visual para o modelo de máquina geométrica. In *XVIII Congresso Regional de Iniciação Científica e Tecnológica em Engenharia*, pages 1–5, Itajaí, SC, Outubro 2003. disponível em CD do congresso.

[4] J. Girard, Y. Lafont, and P. M. Taylor. *Visual Object-Oriented Programming*. Cambridge University Press, Cambridge, 1989.

[5] J. Y. Girard. The system f of variable In *Theoretical Computer Science*, number 45, pages 159–192. 1986.

[6] J. Y. Girard. Linear logic. In *Theoretical Computer Science*, pages 1–102. 1987.

[7] S. McGrath. *XML Aplicações Práticas*. Editora Campus, 1998.

[8] R. Milner. *Communication and Concurrency*. Prentice Hall, 1990.

[9] R. Reiser, M. Cardoso, A. Costa, and G. Dimuro. Utilizando a programação visual no modelo de máquina geométrica. In *CLEI2003-XXIX Conferência Latinoamericana de Informática*, pages 1–10. Universidad Major San Andrés, 2003. meio digital.

[10] R. Reiser and A. Costa. Programming in the quantum geometric machine. In *XXVIII Congresso Nacional de Matemática Aplicada e Computacional*, pages 1–6. SBMAC-RJ, 2005. meio digital.

[11] R. Reiser, A. Costa, and G. Dimuro. First steps in the construction of the geometric machine model. In *TEMA*, volume 1, pages 183–192. 2001.

[12] R. Reiser, A. Costa, and G. Dimuro. The distributed version of the geometric machine model. In *XXVI Congresso Nacional de Matemática Aplicada e Computacional*, pages 163–163. SBMAC-RJ, 2003. meio digital.

[13] R. Reiser, A. Costa, and G. Dimuro. O modelo de máquina geométrica intervalar. In *TEMA*, volume 4, pages 109–118. 2003.

[14] R. Reiser, A. Costa, and G. Dimuro. Programming in the geometric machine. In *Frontiers of Artificial Intelligence*, pages 95–102. IOS Press, Amsterdam, 2003.

[15] R. Reiser, A. Costa, and G. Dimuro. A programming language for the interval geometric machine model. In *Eletronic Notes in Theoretical Computer Science*, volume 84, pages 1–12. 2003.

[16] R. Reiser, A. Costa, and G. Dimuro. The stochastic geometric machine model. In *XXVI Congresso Nacional de Matemática Aplicada e Computacional*, pages 175–176. SBMAC-RJ, 2003. meio digital.

[17] R. Reiser, A. Costa, G. Dimuro, and M. Cardoso. Specifying the geometric machine visual language. In *IEEE Symposium on Visual Languages and Formal Methods*, pages 1–3. 2003.

[18] G. Rossum and L. Drake. Python tutorial, 2003. (disponível via WWW em http://www.pythonorg/doc/current/tut/tut.html).

[19] D. Scott. Some definitional suggestions for automata theory. volume 28, pages 187–212. 1976.

[20] D. West. *Introduction to Graph Theory*. Birkhauser, 1996.

[21] R. Wilson. *Introduction to Graph Theory*. Addison Wesley, 1996.

Correspondencia de Puntos empleando Algoritmos Genéticos para el Registro de Múltiples Vistas de Imágenes de Rango

John Willian Branch
Universidad Nacional de Colombia, Escuela de Sistemas,
Medellin, Colombia
jwbranch@unalmed.edu.co
Germán Sánchez
Universidad Nacional de Colombia, Escuela de Sistemas,
Medellin, Colombia
gsanche@unalmed.edu.co
Sandra Mateus
Universidad Nacional de Colombia, Escuela de Sistemas,
Medellin, Colombia
spmateus@unalmed.edu.co
Idanis Diaz
Universidad de Medellin, Facultad de Ingeniería,
Medellin, Colombia
idanis_diaz@yahoo.es
Flavio Prieto
Universidad Nacional de Colombia, Dept. Eléctrica, Electrónica y Computación,
Manizales, Colombia
fprieto@nevado.manizales.unal.edu.co

Abstract

The registration is a fundamental stage in the reconstruction process 3D, used to match two or more images that can be taken in different moments, from different sensors or from different viewpoints. The genetic algorithms have been applied to the images registration, for their ability to solve problems of optimization. In this paper, a procedure is shown, which is based on genetic algorithms for the registration problem of multiple range images. This procedure is focused, on the problem of obtaining the best correspondence among points through a robust method of search that has more than enough images are partially overlapped. This correspondence set will allow to calculate one transformed which registers the images accurately.

Keywords: Registration, 3D reconstruction, ICP, range images, genetic algorithms.

Resumen

El registro es una etapa fundamental en el proceso de reconstrucción 3D, usado para emparentar dos o más imágenes, que pueden ser tomadas en diferentes momentos, desde diferentes sensores o desde diferentes puntos de vista. Los algoritmos genéticos han sido aplicados al registro de imágenes, por su habilidad para resolver problemas de optimización. En este trabajo, se muestra un procedimiento basado en algoritmos genéticos para el problema de registro de múltiples imágenes de rango. Este procedimiento se enfoca, en el problema de la obtención de la mejor correspondencia entre puntos, a través de un método robusto de búsqueda sobre imágenes que están parcialmente solapadas. Este conjunto de correspondencias, permitirá calcular una transformada que registre con precisión las imágenes.

Palabras claves: Registro, reconstrucción 3D, ICP, imágenes de rango, algoritmos genéticos.

1 Introducción

La reconstrucción 3D es el proceso mediante el cual, objetos reales, son reproducidos en la memoria de un computador, representando algunas de sus características físicas como la dimensión, el volumen y la forma. Esta tarea de reconstrucción de superficies de objetos tridimensionales, a partir de imágenes de rango [14], cubre una serie de etapas: adquisición, registro, integración, segmentación y ajuste, que combinadas llegan a convertir un conjunto de imágenes parciales del objeto en un modelo 3D completo.

El registro es una etapa fundamental en el proceso de reconstrucción 3D, usado para emparentar dos o más imágenes, que pueden ser tomadas en diferentes momentos, desde diferentes sensores o desde diferentes puntos de vista. A través de los años, un amplio rango de técnicas han sido desarrolladas para varios tipos de imágenes y tipos de problemas. Estas técnicas han sido estudiadas independientemente para muchas y diferentes aplicaciones, lo que ha llevado a conformar un buen volumen de investigaciones.

La desalineación que se produce inevitablemente al tomar dos o más imágenes de rango desde puntos de vista diferentes y sin ningún control de las posiciones relativas entre el objeto y el sensor, corresponde al tema central del registro. La información tridimensional capturada en cada imagen está referenciada al sistema de coordenadas de la cámara, el cual por regla general no corresponde al sistema coordenado del mundo. Una vez que la etapa de adquisición culmina con el sensado de la superficie del objeto y se realiza la visualización de las imágenes en un sistema de coordenadas único, se obtienen diferentes vistas que se superponen unas a otras, sin coherencia alguna. El propósito del registro es alinear estas vistas de tal manera que la forma del objeto sea recuperado con la mejor precisión posible.

En el esfuerzo por ajustar las imágenes de manera coherente se hacen evidentes dos situaciones: en la primera, no se tiene certeza de que puntos en el sistema coordenado de una imagen, se corresponde con sus equivalentes en el sistema coordenado de la otra. Este hecho, es conocido como el problema de la correspondencia y constituye la etapa de mayor consumo de tiempo de ejecución del algoritmo. En la segunda, se requiere una transformación sobre la información tridimensional de una de las imágenes de su sistema coordenado al sistema de la imagen que se eligió como referencia, con el propósito de ajustar ambas imágenes utilizando la información común entre ellas. Dada la naturaleza inexacta de los datos y la incertidumbre en el conocimiento de las superficies comunes, el procedimiento de cálculo de esta transformada es iterativo, guiado por las estrategias y métricas elegidas por el investigador. Por ello, el registro es una de las etapas más lentas y delicadas de todo el proceso de reconstrucción 3D, ya que la calidad del alineamiento de las imágenes que en ella se lleva a cabo, determina la calidad del modelo a obtener.

Desde 1992, con la aparición del algoritmo ICP (Iterative Closets Point) [1], son numerosas la variantes que han surgido para mitigar las deficiencias de este método. Este algoritmo planteó un esquema básico para la consecución del alineamiento, minimizando una función de costo basada en la suma de los cuadrados de las distancias entre los puntos de las imágenes; el procedimiento básico involucra la identificación de las características, emparentamiento de las características correspondientes y el alineamiento de estas correspondencias por medio de la evaluación de una métrica para el error [1], [5]. Este método, está también compuesto de dos procedimientos básicos: el primero es encontrar los puntos correspondientes y el segundo, estimar las transformaciones iterativamente para estos puntos hasta satisfacer algún criterio de parada.

Otro enfoque para el registro de imágenes, consiste en determinar un conjunto de correspondencias a través de un proceso de búsqueda, en lugar del enfoque clásico basado en distancia. Este enfoque consiste en encontrar una solución cercana al mínimo global en un tiempo razonable. Una forma de realizar esta búsqueda es mediante los Algoritmos Genéticos (AG).

Los AG han sido aplicados al problema de registro de imágenes; éstos algoritmos son métodos computacionales basados en evolución natural, en los cuales, una población de individuos que representan una posible solución, es evolucionada a través de una sucesión de ciclos de asignación, reproducción, mutación y reemplazo, hasta encontrar la solución deseada [3].

En este trabajo se muestra un procedimiento basado en AG para el problema de registro de múltiples imágenes de rango. Este procedimiento se enfoca, en el problema de la obtención de la mejor correspondencia entre puntos, a través de un método robusto de búsqueda sobre imágenes que están parcialmente solapadas. Este conjunto de correspondencias permitirá calcular una transformada que registre con precisión las imágenes.

Este artículo está organizado de la siguiente manera: En la sección 2, se presenta la revisión de la literatura. En la sección 3, se expone el método de ICP y sus variantes. En la sección 4, se describe la metodología utilizada para realizar la correspondencia de las imágenes de rango usando algoritmos genéticos. En la sección 5, se presentan los experimentos y un análisis de los resultados y en la sección 6, se exponen

las conclusiones.

2 Revisión de la Literatura

Históricamente, la correspondencia de objetos de forma libre usando datos 3D, fue realizada en un principio por Faugueras y su grupo en INRIA [7], donde demostraron una correspondencia efectiva con el timón de un Renault a principios de la década de los 80's. Este trabajo popularizó el uso de cuaterniones para registro de puntos correspondientes mediante mínimos cuadrados. El uso alternativo del algoritmo de descomposición en valores singulares (SVD), no fue tan ampliamente conocido en este tiempo. La limitación primaria de este trabajo fue que partía de la existencia probable de regiones planas razonablemente grandes al interior de una forma libre. Numerosos trabajos posteriores como los de Schwartz [18], Haralick [9], Horn [10], Brou [2] y Taubin [21], constituyen aproximaciones interesantes a la solución del problema del registro, sin embargo, estos trabajos poseen limitaciones en el dominio de las formas de los objetos.

Durante la última década, se han propuesto varias aproximaciones al registro. Estas pueden ser clasificadas básicamente en dos grupos: el registro grueso y el registro fino.

En el registro grueso, la meta es encontrar un conjunto aproximado de transformaciones de registro, sin previo conocimiento de las posiciones espaciales relativas a las vistas. Dentro de los trabajos realizados en el registro grueso se encuentra el de Sappa et al [17], que presentaron un método el cual emplea una técnica de segmentación basada en bordes para guiar el proceso de registro. Chua y Jarvis [6], usaron curvaturas principales y marcos Darboux para calcular características invariantes. También, Feldmar y Ayache [8] propusieron un método para estimar desplazamientos rígidos usando curvaturas principales de superficies.

Stein y Medioni [20] propusieron la estructura Splash, la cual es un mapeo que describe la distribuciones de superficies normales a lo largo de un círculo geodésico. La imagen spin presentada por Johnson y Hebert [11], la cual es un descriptor de forma a nivel de datos, ha sido usada en registro. Lucchese et al [12], explotan la regularidad geométrica obtenida por la transformada de Fourier como un método basado en un dominio de frecuencias para el registro de imágenes de rango. El método DARCES basado en RANSAC de Chen et al [4], es un método robusto basado en la búsqueda exhaustiva que puede chequear todas las alineaciones de datos posibles entre dos conjuntos de datos, para registrar dos vistas parcialmente solapadas.

En contraste a los metodos de registro grueso, los métodos de registro fino se basan en la suposición de que se obtuvo inicialmente una buena transformación inicial. Entonces, las alineaciones precisas se pueden obtener con criterios confiables para medir la calidad de las transformaciones refinadas. Adicionalmente, varias aproximaciones en la literatura proponen el empleo de una combinación de ambas técnicas: un registro grueso, seguido de un registro fino para lograr resultados de registros precisos y automáticos.

Los mejores métodos que se conocen para registro fino de imágenes son las variantes del algoritmo de iteración de punto más cercano (ICP). ICP es un procedimiento iterativo que minimiza cada vez el error cuadrado medio, calculado como la suma de las distancias entre puntos en una vista y los puntos más cercanos, respectivamente en otra vista. Este método fue propuesto por Besl y McKay [1] y constituye el metodo clásico para abordar el problema del registro.

Sin embargo, la desventaja de ICP es que requiere una buena pre-alineación de las vistas para que converja a una solución correcta. Se han propuesto muchas variantes de ICP para sobrellevar estas desventajas. Para el registro con regiones que se solapan sólo parcialmente, se han propuesto heurísticas para ignorar las regiones que no se solapan y obtener consecuentemente transformaciones más efectivas. Las principales diferencias entre estas propuestas, son las funciones de evaluación que miden la calidad de las alineaciones en cada iteración y las reglas de rechazo, tales como, el descarte de puntos de frontera, que podrían conducir a coincidencias falsas. Chen y Medioni [5], desarrollaron un mecanismo que minimiza la suma de las distancias al cuadrado entre puntos en una vista, con respecto a un plano tangente en otra vista. Este acercamiento es relativamente más rápido que el ICP tradicional, y usualmente, los resultados que arroja son mejores si se provee una buena prealineación inicial. Sin embargo, este acercamiento presenta algunas dificultades numéricas en cuanto a las búsquedas, dado que algunas correspondencias pueden no encontrarse.

Zhang [22] propuso un conjunto de modificaciones sofisticadas al ICP y este método ha sido empleado en los sistemas de registro recientes. El método calcula automáticamente un umbral que se utiliza para clasificar un punto como un resultado si su distancia desde el punto correspondiente excede este umbral. Removiendo los puntos obtenidos de los cálculos de registro, se pueden estimar transformaciones más precisas.

Masuda y Yokoya [13] propusieron un método robusto para el registro de un par de imágenes de rango densas, el cual integra ICP, registro aleatorio y el estimador de la menor mediana cuadrada. Ellos también propusieron una modificación del árbol K-d para mejorar la búsqueda y acelerar el método.

Otra aproximación para registrar dos imágenes de rango, es encontrar la transformación geométrica a través de un espacio de búsqueda, más que la búsqueda basada en correspondencias de los métodos basados en ICP. En este caso, la meta es encontrar, en un espacio de búsqueda de transformaciones geométricas, una solución que se pueda emplear para alinear precisamente dos vistas. Una manera razonable para realizar esta búsqeda es a través del empleo de técnicas eficientes de optimización estocástica, tales como los Algoritmos Genéticos. Esta aproximación, generalmente se considera para proveer registro grueso. Sin embargo, se pueden combinar diferentes operadores, tales como búsquedas heurísticas locales, para obtener alineaciones precisas durante el proceso de convergencia. De esa manera, los resultados del registro final pueden compararse con aproximaciones tradicionales al registro fino.

Brunnstrom y Stoddart [3] propusieron un método en el que se integra el método clásico ICP con un algoritmo genético para acoplar superficies de forma libre. En esta propuesta, inicialmente se obtiene un alineamiento con un algoritmo genético, que luego es refinado con el ICP. El problema central hacía el cual Brunnstrom y Stoddart dirigen su propuesta, es el encontrar un conjunto de puntos correspondientes entre las dos vistas, para ello, se toman muestras densas en ambas vistas y se procede a hacer la búsqueda con un algoritmo genético que va asociando puntos entre las vistas, guiado por una función de aptitud que va contando el número de buenas correspondencias utilizando las invariantes de traslación y rotación, tales como, la orientación relativa de las superficies normales y la distancia relativa entre puntos. En esta propuesta, un cromosoma representa una asignación de puntos de ambas vistas.

Robertson y Fisher [15] propusieron un algoritmo genético paralelo, con el cual se logra reducir el tiempo computacional empleado, pero las soluciones no son más precisas que las obtenidas con el primer método. En esta propuesta los individuos de la población son vectores conformados por seis parámetros, que representan una transformada. La función de aptitud empleada es una medida del error medio de las dos vistas registradas, luego de aplicar la transformada representada por cada cromosoma de la población.

Silva, Bellon y Boyer [19], proponen un método para el registro de imágenes de rango, haciendo dos contribuciones claves: la hibridación de un algoritmo genético con el método de optimización heurístico de Ascenso por la Colina y una medida de desempeño de interpretación de la superficie, diferente a la métrica clásica basada en el cálculo del error cuadrático medio entre los correspondientes puntos de dos imágenes, luego del registro. La medida de desempeño propuesta en este trabajo, consiste en calcular la fracción de puntos que quedan ínterpenetrados en la vista A y en la vista B después del registro. Este método se especializa en buscar los parámetros de una transformada constituida por seis valores, tres parámetros de rotación y tres para traslación.

La revisión de la literatura sobre la problemática del registro revela los numerosos intentos por resolver dicho problema. Entre ellos el algoritmo ICP tiene un lugar destacado, a pesar de sus serias limitantes. Por ejemplo, pese a la satisfactoria convergencia del ICP, sólo se puede garantizar cuando una de las imágenes es un subconjunto de la otra; sin cumplir esto, se pueden alcanzar alineamientos erróneos. Aunque, el ICP es eficiente, con un promedio de complejidad de O(n log n) donde n es el número de puntos en la imagen, el algoritmo converge monótonamente a un mínimo local. Otra desventaja del ICP, es que requiere de una buena pre-alineación de las vistas para converger a una solución correcta. Algunas variantes del ICP han sido propuestas para tratar de solucionar estas limitaciones.

Existe una gran dificultad para comparar las diferentes propuestas, debido a que cada investigador utiliza una base de imágenes diferente, lo que impide una comparación de los resultados de las métricas que cada uno implementa.

Ocasionalmente, la estrategia para prealinear las imágenes puede guiar el proceso a una convergencia que obtiene una solución errónea. Por otro lado, las estrategias que exploran exhaustivamente el espacio de correspondencias y transformaciones son computacionalmente muy costosas. Aunque se realice un rechazo de los parejas erróneas esto no es un parámetro suficiente para garantizar un adecuado ajuste.

La evaluación de la precisión del ajuste es otro ítem que requiere atención; el camino más fácil es la comparación del modelo obtenido con otro de referencia ya sea sintético o real.

Otro aspecto que afecta el desempeño del método, es el tamaño de las imágenes. Los escáneres modernos pueden ofrecer elevadas resoluciones por lo cual la densidad de información en las imágenes es alta. Por esta razón, se proponen estrategias de submuestreo para reducir el número de puntos correspondientes para guiar el registro, el muestreo uniforme aleatorio y el muestreo uniforme en el espacio de las normales, [16].

Otra consideración en el tema de registro son las reglas de rechazo, esto es, las estrategias para depurar los apareamientos descartando aquellos incorrectos. Una de las principales reglas es la exclusión de los puntos de frontera, su aplicación es poco costosa y excluye regiones que no están solapadas.

Se debe considerar, la combinación de métodos paramétricos y de minimización, a partir de un registro inicial aceptable, dado que generalmente los métodos paramétricos ofrecen ventajas en cuanto a la velocidad de convergencia y los de minimización alcanzan niveles superiores de precisión.

3 El algoritmo ICP y sus variantes

Un conjunto de puntos se mueve de manera rígida, de tal forma que estén alineados de la mejor manera posible con el correspondiente modelo CAD, a través del siguiente procedimiento interativo:

En el primer paso de cada iteración, por cada punto de los datos del objeto, se calcula el punto más cercano en la vista de referencia del objeto. Esta es la parte del algoritmo que consume más tiempo y puede ser implementada eficientemente, por ejemplo, usando una estructura de datos octree. Como resultado del primer paso, se obtiene una secuencia de puntos $Y = (y_1, y_2, ...)$ de los puntos de la forma más cercanos a la secuencia de datos $X = (x_1,x_2, ...)$. Cada punto x_i corresponde al punto y_i con el mismo índice.

En el segundo paso de cada iteración, el movimiento rigido M se calcula tal que los puntos de datos movidos M(x_i) sean los más cercanos a sus puntos correspondientes y_i donde la función objetivo a minimizarse es:

$$\sum_{i=1}^{n} \|y_i - M(x_i)\|^2$$

Este problema de mínimos cuadrados se puede resolver explícitamente. La parte translacional de M trae el centro de masa de X al centro de masa de Y. La parte rotacional de M se puede obtener como el vector propio unitario que corresponde al máximo valor propio de una matriz simétrica 4x4. El vector propio solución no es otra cosa que el cuaternión unitario de la parte rotacional de M.

Después de este segundo paso, las posiciones de los puntos de datos se actualizan vía $X_{nueva} = M(X_{vieja})$. Ahora los pasos 1 y 2 se repiten, siempre usando los datos de los puntos actualizados, mientras el cambio en el error cuadrático medio se mantenga por debajo de un umbral preseleccionado. El algoritmo de ICP siempre converge monotónicamente a un mínimo local, ya que el valor de la función objetivo siempre se decrementa en los pasos 1 y 2.

Una de las variantes más importantes del ICP, consiste en modificar la etapa de selección de los puntos, de tal forma que la distribución de las normales de los puntos seleccionados sea la mayor posible. Esta idea fue motivada sobre el hecho, de que pequeños detalles representados por puntos con alta variación de sus normales, son muy importantes para el correcto alineamiento de las imágenes; en contraste a esto, los métodos basados en selección aleatoria no podrían garantizar la selección de estos puntos.

El método de selección en el Espacio de las Normales fue presentado por Rusinkiewicz [16]; este consiste, en una variación del muestreo aleatorio y busca lograr equidad estadística en el muestreo para los tres ejes coordenados. La orientación de la normal unitaria asociada al plano que forma cada punto con sus vecinos más próximos, varía de acuerdo con los cambios de forma en la imagen. El muestreo aleatorio en el espacio de normales, toma un conjunto de puntos de la imagen de forma aleatoria pero uniforme en las orientaciones de las normales asociadas a los puntos. Sea X una superficie y x_i los puntos que la representan, para cada x_i existe un vector normal unitario n_i con componentes n_ix, n_iy, n_iz; la componente con mayor magnitud determina la tendencia de la orientación de la normal. Posteriormente, se separan los puntos x_i en grupos por orientación y se toman los datos aleatoriamente en cada grupo, conservando la secuencia de los ejes x, y y z, asegurando de esta manera, que los datos están proporcionados en el sentido de su orientación. Una vez se obtiene las dos muestras MA y MB de cada imagen, se procede a buscar los puntos correspondientes entre cada punto de MA y su pareja afín en MB.

4 Método de Correspondencia de Puntos de Imágenes de Rango empleando Algoritmos Genéticos

El principio general de un algoritmo genético es someter a un proceso de evolución una población de individuos codificados como cromosomas, los cuales representan posibles soluciones de un problema de búsqueda.

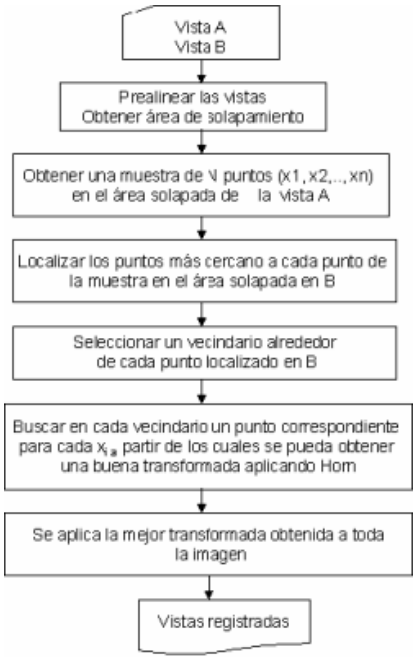


Figure 1: Esquema general del método de registro propuesto

Durante la evolución, a cada individuo se le asigna un valor de aptitud obtenido de una función definida específicamente para el problema a solucionar. Esta función, llamada función de aptitud, debe ser diseñada de tal forma que favorezca a los individuos más aptos o adecuados como solución del problema. La aptitud asignada a cada individuo es tenida en cuenta para seleccionar los progenitores a participar en el proceso de la reproducción, el cual consiste en intercambiar el material genético o contenido de un par de individuos seleccionados para generar dos nuevos individuos o dos nuevas posibles soluciones del problema, que de acuerdo a un mecanismo de reemplazo son incorporadas en la población. Los nuevos individuos descendientes se someten además a un proceso de mutación que consiste en una perturbación aleatoria de su material genético, con el objetivo de dar variabilidad y enriquecer la exploración de las posibles soluciones del problema, representadas como cromosomas. Finalmente, después de llevarse a cabo un determinado número de ciclos de asignación de aptitud, reproducción, mutación y reemplazo, llamados generaciones, se escoge como la mejor solución para el problema, el individuo con mejor aptitud.

Los algoritmos genéticos han sido aplicados en el problema de registro; sin embargo, la complejidad del espacio de búsqueda, ha sido uno de las principales dificultades para las propuestas realizadas. A continuación se plantea y describe una propuesta para utilizar algoritmos genéticos en el problema de registros de vistas de imágenes rango prealineadas. La propuesta se basa en la búsqueda de un conjunto de puntos que al ser tomados como entrada para el método de Horn, se obtenga una buena transformada que permita integrar las imágenes con un margen de error muy pequeño. En la Figura 1, se muestra un esquema general del método propuesto.

Las vistas a registrar son prealineadas con el objetivo de obtener un área de solapamiento inicial en ambas imágenes. Como se puede observar, en los pasos siguientes para cada punto de una muestra de tamaño N tomada en el área de solapamiento de una de las vistas, se busca un punto correspondiente alrededor de los puntos más cercanos en la otra vista a registrar. Esta búsqueda se realiza, porque no siempre las mejores parejas de puntos para obtener una transformada por medio del método de Horn son los puntos con menor distancia en un área de solapamiento. Dos vistas podrían estar mal alineadas y presentar puntos con distancias muy pequeñas; sin embargo, al unir las vistas utilizando estos puntos como guía podría conducirse a un mal registro de ellas. Las imágenes prealineadas inicialmente podrían quedar algo trasladadas y los puntos correspondientes con los que mejor se acoplarían las vistas al aplicar una transformada, podrían estar muy cercanos a los puntos con mínima distancia.

Dadas dos imágenes de rango A y B, donde A es la imagen modelo y B es la imagen a registrar, la búsqueda de los mejores puntos en A que se acoplan con una muestra de puntos seleccionada en B, es realizada por un algoritmo genético, el cual está diseñado como sigue:

Muestreo: Consiste en seleccionar aleatoriamente N puntos pertenecientes al área solapada en B y determinar, para cada uno de estos, un subconjunto de puntos o subdominio en A. Los subdominios contienen m puntos cercanos al punto más cercano en A para cada punto en B. Este enfoque de subdominios, reduce el espacio de búsqueda y mejora la eficiencia global del algoritmo.

La determinación de los dominios, posee un paso crítico computacionalmente; este es, la búsqueda del punto más cercano en A a cada uno de los puntos de la muestra seleccionada en B, debido a que esto implica calcular y comparar las distancias a todos los puntos que forman el área solapada en A. Esta búsqueda es mejorada con la implementación de una estructura de árbol K-d. La Figura 2 muestra gráficamente la determinación de un subdominio.

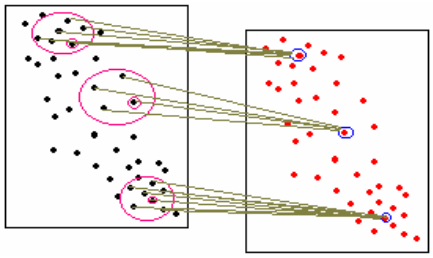


Figure 2: Determinación de sub dominios

Esquema de representación: Consiste en un cromosoma de tamaño N, esto es, a cada uno de los puntos de la muestra seleccionada en la vista B le corresponde un gen del cromosoma. Cada gen contiene un índice que identifica un punto dentro del vecindario correspondiente a dicho punto, definido en la vista A. La Figura 3, ilustra esta representación.

El gen 1 corresponde al primer punto de la muestra, el gen 2 al segundo punto de la muestra y así sucesivamente hasta el N-ésimo punto de la muestra tomada en la vista B. Por ejemplo, en la Figura 3 el gen 1 contiene el valor 12, lo cual significa que el punto 12 se encuentra dentro del subdominio correspondiente al primer punto de la muestra en B, el 25 es un índice de un punto de la vista A, perteneciente a un vecindario de puntos cercanos al punto 2 de la muestra, y así sucesivamente. Cada punto de la muestra tomada en la vista B tiene definido un vecindario de puntos cercanos en la vista A, del cual el respectivo gen tomará valores.

Función de aptitud: La función de aptitud para esta propuesta consiste en medir el error promedio entre los puntos de las áreas de solapamiento originadas en el registro de las vistas. Cada individuo, que puede ser visto como un conjunto de puntos con sus respectivas parejas, es traducido a una transformada por medio del método de Horn. La transformada es aplicada a las dos vistas y el error medio de este registro es asignado como la aptitud del individuo, entre más pequeño sea este error, el individuo será mucho mejor:

$$\epsilon = \frac{\sum_{i=1}^{N} \sqrt{(P_i - R_i)^2}}{N}$$

P, es cada punto en el área de solapamiento en la vista A, obtenida al aplicar cada transformada y R es cada punto en el área de solapamiento en la vista B, después de aplicada la transformada.

Operadores Genéticos: : La propuesta presentada para el registro de dos vistas aplica un operador de cruzamiento simple con un solo punto de corte, en el que se intercambia el contenido genético de los

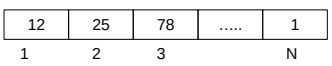


Figure 3: Esquema de representación

Figure 4: Cruce con un solo punto de corte.

progenitores a cada lado del punto de corte para generar dos nuevos individuos (Ver Figura 4). Por su parte, el operador de mutación, simplemente varía la información de cada gen según la probabilidad de mutación, teniendo en cuenta los vecindarios definidos para cada punto representado. Es decir, si al gen i que representa el i-ésimo valor de la muestra tomada en la vista B, le corresponde ser mutado, se selecciona al azar un respectivo punto en el vecindario definido para este punto en la vista A y se cambia por el anterior valor.

El muestreo de N puntos en la vista B, sólo se realiza una vez durante todo el algoritmo genético y por lo general N es un número pequeño, lo que quiere decir que el algoritmo genético descrito, se especializa en la búsqueda de los puntos que mejor se acoplen con una muestra pequeña de puntos fija. El prealineamiento de las imágenes nos permite reducir el espacio de búsqueda para este procedimiento.

5 Experimentos y Resultados

5.1 Configuración del experimento

5.1.1 Hadware y librerías

Todas las pruebas fueron realizadas utlizando una computadora con procesador de 3.0G, memoria RAM de 1.0G corriendo bajo el sistema operativo Microsoft XP. Las implementaciones de los modelos fueron realizados en C++ y se programó un motor gráfico en OpenGL, para obtener la representación gráfica de las imágenes. Los datos utilizados fueron obtendidos con un sensor Kreon en el Advanced Man-Machine Interface Laboratory - Departament of Computing Science, Canadá.

5.1.2 Parametrización del Algoritmo Genético (AG)

La ejecución de un AG, exige asignar valores apropiados a un conjunto de parámetros que garanticen su buen funcionamiento. Dentro de dicho conjunto de parámetros se encuentran, entre otros, el tamaño de la población, la longitud de los individuos y las probabilidades asignadas a cada uno de los operadores genéticos.

Con el objetivo de determinar los valores adecuados de las probabilidades de los operadores genéticos, para el problema de registro de imágenes de rango, se emplearon una decena de pares de imágenes de rango reales, con una variación de las probabilidades de los operadores de cruce y mutación, en intervalos de 10% y promediando los resultados obtenidos para cada una de las posibles combinaciones de porcentaje de estos. En la Figura 5 se muestran los promedios de los resultados para cada combinación, después de 100 generaciones de evolución del AG.

Los resultados con menor promedio se consiguieron con una probabilidad del 40% para el operador de cruce y del 70% para el operador de mutación. El tamaño de la población fue establecido en 100 individuos, cada uno de los cuales está compuesto por 10 parejas de puntos. La Figura 6 muestra los valores de los parámetros de configuración para el AG empleado en los diferentes experimentos de este trabajo.

5.1.3 Calibración de los modelos

Debido a que el modelo de AG trabaja en el problema específico, de encontrar el mejor emparentamiento de puntos que permitan encontrar una transformada que registre correctamente un par de imágenes y con el objetivo de validar el correcto funcionamiento del método ICP e ICP+AG , se realizaron pruebas sobre datos sintéticos, en los cuales se aseguraba una correspondencia punto a punto entre las imágenes, que garantiza la existencia de una solución única del problema.

MUTACIÓN \ CRUCE	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.1742	0.1968	0.1984	0.1899	0.1947	0.1980	0.1882	0.1859	0.1971
0.2	0.1925	0.1892	0.1949	0.1653	0.1858	0.1752	0.1842	0.1771	0.1819
0.3	0.2044	0.1991	0.1873	0.1841	0.1879	0.1915	0.1842	0.1895	0.1997
0.4	0.1845	0.1766	0.1876	0.1977	0.1961	0.1915	0.1770	0.1699	0.1891
0.5	0.1899	0.1885	0.1929	0.1902	0.1799	0.1863	0.1890	0.1938	0.1792
0.6	0.1967	0.1897	0.1855	0.1650	0.1990	0.1889	0.1930	0.1927	0.1722
0.7	0.1950	0.1846	0.1906	0.1608	0.1855	0.1937	0.1968	0.1863	0.1611
0.8	0.2017	0.1890	0.1968	0.1969	0.1894	0.1921	0.1750	0.1942	0.1834
0.9	0.1825	0.1912	0.2035	0.1870	0.1860	0.1886	0.1877	0.1858	0.1794

Figure 5: Promedio de los resultados obtenidos para cada combinación de porcentajes asignados a los operadores genéticos

PARÁMETRO	VALOR
Número de Generaciones	100
Tamaño de la Población	100
Tamaño del Individuo	10 Parejas
Probabilidad de Cruce	40%
Probabilidad de Mutación	70%

Figure 6: Valores de configuracion del AG

El experimento consistió, en la selección de una imagen de rango de un objeto real escaneado, que sirvió como imagen modelo del proceso de registro. La imagen a registrar fue generada mediante la variación de los puntos de una copia de la imagen modelo, con una matriz de transformada conocida, generando así, un par de imágenes en las cuales se garantizaba la existencia de una correspondencia única punto a punto.

Este experimento fue realizado con un AG que evolucionó 100 generaciones, cuyos parámetros fueron definidos por las pruebas experimentales de la sección anterior. El valor de convergencia del AG fue establecido en $1x10^{-10}$, un valor muy bajo para permitir que se encontrara la mejor solución posible durante la totalidad de las generaciones. La matriz de transformación utilizada para generar la imagen a registrar, se construyó en tres casos diferentes: en el primer caso, se varió la imagen en cada uno de los ángulos de rotación; en el segundo caso, se variaron los parámetros de traslación; y finalmente, se generó una matriz con la combinación de los dos casos anteriores (Ver Figura 7). En la Tabla 1, se muestran los valores con los que se construyeron las transformadas para el tercer caso expuesto.

Los resultados de estos experimentos mostraron que los métodos ICP e ICP+AG, son capaces de encontrar una solución y permitieron registrar el par de imágenes dado en cada uno de los diferentes casos. El error del registro final de las imágenes mediante los dos métodos, mostró una convergencia al mínimo global como se muestra en la Tabla 1. Además el comportamiento de los modelos mostró una convergencia muy rápida hacia la solución; en ningún caso la convergencia se alcanzó después de 35 iteraciones o generaciones.

Table 1: Variación de los parámetros de rotación y traslación

CASO 3	E	Rx	Ry	Rz	Tx	Ty	Tz	Error
Transformación	1	0.070	0.05	0.03	1	3	1	
ICP	0.999	0.067	0.058	0.043	1.251	2.974	1.524	2.52E-08
ICP+AG	0.999	0.069	0.055	0.027	1.025	3.036	0.856	6.52E-10

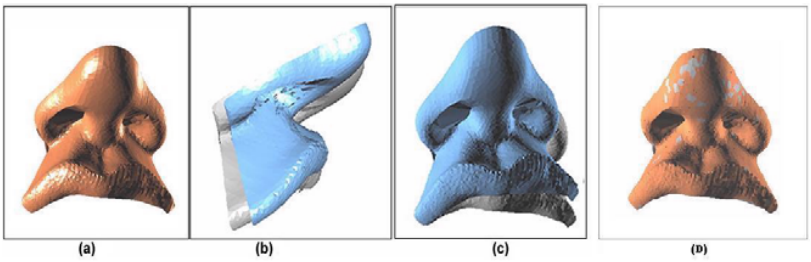


Figure 7: Rotación y traslación de los datos sintéticos, a) imagen modelo, b) y c) imagen modelo e imagen a registrar, d) resultado final

5.2 Análisis de Resultados

En esta sección, se analizan los resultados obtenidos por el método ICP y una variante basada en orientación de las normales con el método propuesto de AG, sobre un ejemplo de aplicación real. El objeto registrado corresponde a una máscara, compuesta por 8 imágenes de rango obtenidas empleando el sensor Kreon.

El proceso de registro se realizó, registrando cada una de las imágenes con la fusión de las imágenes anteriormente registradas. En la Figura 8, se muestra la secuencia del proceso de registro, asi como el objeto final registrado, obtenido para cada imagen del objeto máscara. Los resultados graficados corresponden a los obtenidos con el modelo de AG durante 100 generaciones.

Como se puede ver en la Figura 9, en todos los casos se obtuvo un menor error de registro con el modelo AG (Error Promedio=0.1011), frente al modelo clásico ICP (Error Promedio=0.1196) y la variante ICP+Normales (Error Promedio=0.1104); cada modelo fue ejecutado con el mismo número de iteraciones o generaciones y los valores para los parámetros del modelo AG fueron establecidos mediante las pruebas realizadas en las secciones anteriores. Sin embargo, las diferencias de tiempo entre los diferentes métodos y el AG, es significativa para estas pruebas en las que las imágenes contienen en promedio 35000 puntos. Los métodos clásicos, tardaron en promedio 1.5 minutos, para registrar cada par de vistas, mientras que, el modelo de AG, tarda en promedio 7 minutos por cada par de vistas.

6 Conclusiones

Se ha propuesto un método semiautomático para el registro de múltiples vistas de imágenes de rango con bajo solapamiento, el cual está en capacidad de encontrar un registro adecuado, sin la necesidad de un prealineamiento fino entre las imágenes. El método está basado en algoritmos genéticos, para realizar la búsqueda de las mejores correspondencias de un conjunto muestreado de puntos, a partir, de un enfoque basado en subdominios, el cual reduce el espacio de búsqueda del algoritmo genético, lo que conlleva a una mejora de la eficiencia global del algoritmo.

La comparación de los resultados obtenidos mediante los diferentes experimentos realizados, muestra una convergencia más precisa con el método propuesto (ICP+AG), que el método clásico ICP y una de sus variantes (ICP+Normales). Sin embargo, el método propuesto emplea más tiempo computacional en encontrar la solución.

Se estableció un esquema para la reconstrucción del modelo 3D que permite la obtención de modelos parciales y volumétricos, donde el resultado final se puede presentar como combinaciones de mallas triangulares o nube de puntos y el error final está definido como el promedio de los errores parciales obtenidos en el registro de cada par de vistas.

Como trabajo futuro, se propone explorar una versión paralela para reducir el costo computacional del método propuesto.

References

[1] P. Besl. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell*, 14, 1992.

[2] P. Brou. Using the gaussian image to find the orientation of an object. *Int. Journal Robotics*, 3, 1983.

[3] K. Brunnstrom. Genetic algorithms for free-form surface matching. 1996.

[4] C. Chen. Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(11), 1999.

[5] Y. Chen. Object modeling by registration of multiple range images. *Image and Vision Computing*, 10, 1992.

[6] C.S. Chua. 3d free-form surface registration and object recognition. *International Journal of Computer vision*, 17, 1996.

[7] O.D Faugueras. The representation, recognition, and locating of 3d objects. *International Journal of Robotics Research*, 5, 1986.

[8] J. Feldmar. Rigid affine and locally affine registration of free-from surfaces. *International Journal of Computer Vision*, 18(2), 1996.

[9] R. Haralick. Pose estimation from corresponding point data. *Machine Vision for Inspection and Measurement*, 1989.

[10] B. Horn. Closed-form solution of orientation using unit quarternions. *Journal of Optica Society of America*, 4, 1987.

[11] A. Johnson. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell*, 21, 1999.

[12] L. Lucchese. A frequency domain technique for range data registration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(11), 2002.

[13] T. Masuda. A robust method for registration and segmentation of multiple range images. *Computer Vision and Image Understanding*, 61(3), 1995.

[14] A. Myers. Introductory literature review surface reconstruction from three dimensional range data. Technical report, The University of Adelaidey, Department of Computer Science, 1999.

[15] C. Robertson. Parallel evolutionary registration of range data. *Computer Vision and Image Understanding*, 87, 2002.

[16] S. Rusinkiewiczs. *Real-time Acquisition and Rendering of Large 3D Models*. PhD thesis, Stanford University, 2001.

[17] A.D. Sappa. Range image registration by using an edge-based representation. 2001. http://www.siggraph.org.

[18] J. Schwartz. Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. *Int. Journal Robotics*, 6, 1987.

[19] L. Silva. Precision range image registration using a robust surface interpretation measure an enhanced genetic algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(05), 2005.

[20] F. Stein. Structural indexing: Efficient 3-d object recognition. *IEEE Trans. Pattern Anal. Mach. Intell*, 14, 1992.

[21] G. Taubin. Algebraic nonplanar curve and surface estimation in 3-space with applications to position estimation. Technical report, Brown University, 1988.

[22] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2), 1994.

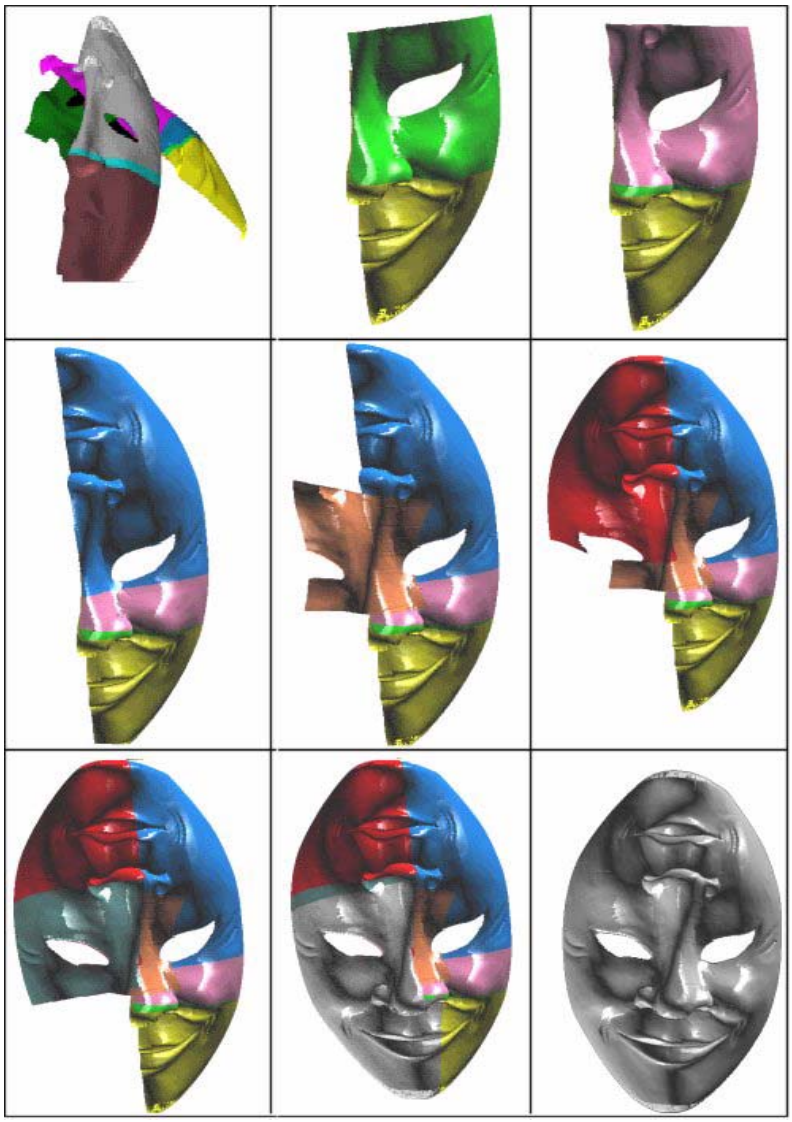


Figure 8: Secuencia del proceso de registro de la Máscara

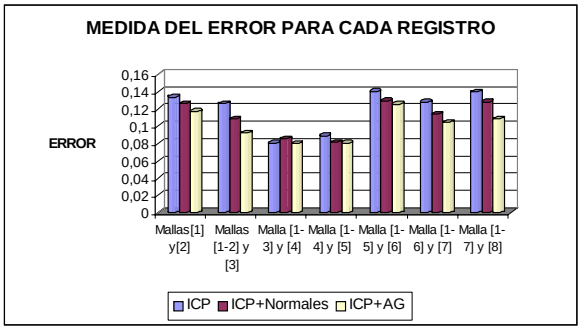


Figure 9: Error de registro por modelo de cada par de imágenes

An MPEG-7 Learning Space for Semantic Image Classification

Andres Dorado
Pontificia Universidad Javeriana, School of Engineering,
Cali, Colombia
Queen Mary, University of London, Dept. of Electronic Engineering,
London, UK
andres.dorado@elec.qmul.ac.uk

and

Ebroul Izquierdo
Queen Mary, University of London, Dept. of Electronic Engineering,
London, UK
ebroul.izquierdo@elec.qmul.ac.uk

and

Witold Pedrycz
Univesity of Alberta, Dept. of Electrical and Computer Engineering,
Edmonton, Canada
pedrycz@ece.ualberta.ca

Abstract

This paper is concerned with the use of MPEG-7 visual descriptors for defining a feature space as part of a generic framework for semantic image classification. Feature vectors consist of colour and texture descriptors, whose elements are aggregated in such a way that descriptor overriding is avoided, original topology is preserved and vector dimensionality controlled. Clustering analysis is used to evaluate the quality of proposed vector structure regarding to its ability to discriminate examples from different classes. Thus, the feature space is partitioned into semantically meaningful groups whose interpretation may relate to some description task pertaining to the image content.

Keywords: Image Classification, MPEG-7, Feature Selection, Clustering.

1 INTRODUCTION

The ongoing interest in high-level image classification can be evidenced by standardization activities such as MPEG-7 and its Visual Core Experiments as well as the recent Still Image Search project (JPSearch) [ref www.jpeg.org/]. MPEG-7 descriptors are equipped with a well-defined syntax and semantics that facilitate representation of image abstractions.

High-level (or semantic) image classification relates the problem of assigning an image to one or more specified categories based on interpretation of its content. In designing the classifier, our interest turned to the use of MPEG-7 descriptors for defining a feature space. Specifically, we are concern on the structure of the feature vectors in order to accomplish class separability and generalization. Separability is satisfied when descriptions are very different for images in different classes. On the other hand, generalization is achieved when descriptors contain very similar values for images in the same class.

The problem of combining descriptions has been tackled using either multi-feature vectors or multiple feature spaces [19][7]. The approaches using multi-feature vectors break down into high-dimensional space, which drastically reduce efficiency of storage and retrieval. [8] introduced an unsupervised feature extraction technique applying an orthogonal subspace projection that reduces the data dimensionality. [10] used tree structured wavelet decomposition for dimensionality reduction of texture features. The work presented by [20] reduces dimensionality preserving the local topology of the original space. [2] proposed an adaptive clustering method to select representative features.

On the other hand, approaches apply on multiple feature spaces require an effective method to integrate the matching results form each space. [17] detailed a completed taxonomy of the different searching and matching problems on multi-feature spaces. Furthermore, [11] proposed a framework that integrates a number of parallel tree structured self-organising maps.

We propose a feature vector structure based on specific elements from MPEG-7 colour and texture descriptors. Descriptor elements are organized into feature vectors aiming at (1) preserve the descriptors' semantics, (2) obtain an abstract representation of the image content close to the visual perception of the beholder, (3) avoid description overriding, and (4) control the vector dimensionality. Clustering analysis is used to evaluate the quality of proposed structure regarding to its ability to discriminate examples from different classes.

The paper is organized as follows: Sect. 2 introduces the Multimedia Content Description Interface, MPEG-7. Sect. 3 summarises some considerations in the construction of the feature space and details the components of feature vectors using the proposed structure. Sect. 4 shows experimental results. Concluding remarks are given in Sect. 5.

2 MPEG-7

MPEG-7, formally known as Multimedia Content Description Interface is an ISO/IEC standard developed by the Moving Picture Experts Group (MPEG) for description and search of audio and video content [ref www.chiariglione.org/mpeg/]. In contrast with the early standards known as MPEG-1, MPEG-2, and MPEG-4 that are focused on coding and representation of audio-visual content, MPEG-7 moves forward and becomes more general by embracing description of description of multimedia content [12].

MPEG-7 has emerged as a cornerstone of the development of a wide spectrum of applications dealing with audio, speech, video, still pictures, graphics, 3D models, and alike. In a nutshell, the MPEG-7 environment delivers a comprehensible metadata description standard that is interoperable with other leading standards such as SMPTE Metadata Dictionary, Dublin Core, EBU P/Meta, and TV Anytime; refer to www.ebu.ch/trev_284-mulder.pdf. Initially, MPEG-7 was focused more on web-based applications and annotation tools (e.g. [9][14][18]). Nowadays, it is being drifted to other domains such as education, video surveillance, entertainment, medicine and biomedicine.

The ultimate objective of MPEG-7 is to provide interoperability among systems and applications used in generation, management, distribution, and consumption of audio-visual content descriptions (see Fig. 1. Such descriptions of streamed (live) or stored on various media help either users or applications in identifying, retrieving, or filtering essential audio-visual information, cf. [12][1].

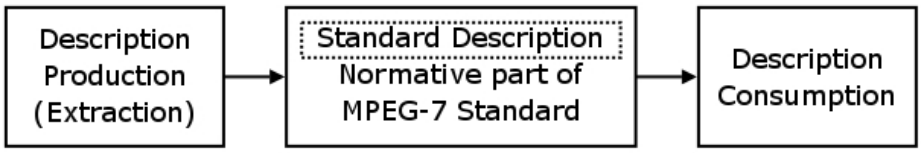


Fig. 1: Scope of MPEG-7 [1]

MPEG-7 specifies standardised Descriptors and Description Schemes for audio and video, as well as integrated multimedia content. Also standardised is a Description Definition Language that allows new Descriptors and Description Schemes to be defined.

MPEG-7 descriptors define syntax and semantics of features of audio-visual content. MPEG-7 allows these descriptions at different perceptual and semantic levels. At the lowest abstraction level, such descriptors may include shape, texture, and colour. At the highest abstraction level, they may include events, abstract

concepts, and so forth. Tab. 1 presents a list of the basic MPEG-7 visual descriptors applicable to still images.

Tab. 1: List of basic MPEG-7 visual descriptors used to characterize still images [13]

Colour Descriptors	**Texture Descriptors**	**Shape Descriptors**
Colour Layout	Texture Browsing	Region–based Shape
Colour Structure	Homogeneous Texture	Contour–based Shape
Dominant Colour	Edge Histogram	
Scalable Colour		

During the collaborative phase of the MPEG-7 standard, descriptors were incorporated into a common model called the eXperimentation Model or XM, which constitutes a draft of the standard itself [ref www.chiariglione.org/mpeg/]. Test conditions and criteria to assess those descriptors were well defined. In the case of visual descriptors, tests were oriented to evaluate retrieval performance. It has motivated complementary studies to analyse aspects such as data quality of the extracted descriptions [4][5] or computational complexity [15][16].

3 MPEG-7 FEATURE SPACE

Proposed feature space is built using a number of selected histogram-based MPEG-7 colour and texture descriptors [13]. Treating such histograms as feature vectors

$$\mathbf{x} = [x_1, x_2, \ldots, x_p]^T \quad , \tag{1}$$

where $x_i \in \Re;\ 1 \leq i \leq p$, they can be organized into a feature space

$$X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_X}\} \ , \tag{2}$$

considering contributions of each descriptor element regarding its semantics and capabilities to represent the image content. It is worth stressing that the problem of feature selection is fundamental to pattern recognition [3], it is to say the assignment of an object to one or several specified categories.

There is a vast array of methods including such prominent approaches as principal component analysis (PCA), independent component analysis (ICA) and alike [6] commonly used to select features and reduce dimensionality of feature space. A drawback of those methods in working with MPEG-7 descriptors is that they do not relate directly the original topology and consequently the interpretation of transformed features becomes less intuitive.

We proposed a feature vector structure that combines efficiently features extracted from MPEG-7 colour and texture descriptors. The structure keeps the original topology to preserve syntax and embedded semantics of MPEG-7 descriptors. It selects descriptor elements avoiding not only description overriding but also controlling vector dimensionality. Quality of feature vectors is evaluated in terms of class separability and generalisation.

3.1 Colour and Texture Descriptors

As aforementioned, the feature space used in this study is formed by making use of two types of visual descriptors, namely colour and texture. Then, image content is captured by combining information extracted from local and global distributions (histogram) of colour, edges, and texture. For the sake of completeness, next sections include the components of the entire feature vector structure along with a brief description of their syntax and semantics.

3.1.1 Colour Layout Descriptor

This descriptor captures the spatial distribution of colour. It applies the discrete cosine transform (DCT) to representative colours in a 8×8 grid and encodes the resulting coefficients. A picture is divided into 64

(8×8) blocks and their average colours are derived. The colour space adopted for CLD is YCrCb. Fig. 2 shows a sample of Colour Layout Description in XML format.

```
<Descriptor xsi:type="ColorLayoutType">
    <YDCCoeff>6</YDCCoeff>
    <CbDCCoeff>3</CbDCCoeff>
    <CrDCCoeff>3</CbDCCoeff>
    <YACCoeff6>24 18 31 21 16 15</YACCoeff6>
    <CbACCoeff3>32 31 14</CbACCoeff3>
    <CrACCoeff3>18 15 15</CrACCoeff3>
</Descriptor>
```

Fig. 2: Sample of Colour Layout Description in XML format

3.1.2 Colour Structure Descriptor

This descriptor scans an image using a sliding window approach to capture localized colour distributions. Area of the window is defined by an $n \times n$-pel structuring block, which by default uses $n = 8$. A 1×1-pixel window reduces description to a standard colour histogram.

The histogram summarizes the number of times colours are reported as occurring within the window. A colour map in the Hue-Min-Max-Diff (HMMD) colour space defined by MPEG and non-uniform quantisation determines the maximum number of bins.

Fig. 3(a) presents the mapping from RGB to HMMD space. The Hue channel has the same meaning as in the HSV space. If `Max` == `Min` Hue is undefined (achromatic colour); otherwise it is computed as indicated in Fig. 3(b).

```
Max   =  max(R, G, B)
Min   =  min(R, G, B)
Diff  =  Max - Min
```

(a) Min-Max-Diff

```
if (Max == R && G > B) Hue = 60*(G-B)/Diff
else if (Max == R && G < B) Hue = 360 + 60*(G-B)/Diff
else if (G == Max) Hue = 60*(2.0 + (B-R)/Diff)
else Hue = 60*(4.0 + (R-G)/Diff)
```

(b) Hue

Fig. 3: Computing HMMD space from RGB values

Fig. 4 shows a sample of Colour Structure Description in XML format.

```
<Descriptor xsi:type="ColorStructureType" colorQuant="4">
    <Values>0 0 0 0 0 0 0 0 33 40 1 21 0 0 0 0 0 2 0 0 0 0 0 0 61 88
            85 60 2 1 5 80 78 41 54 19 32 17 1 92 100 72 61 16 8 6
            42 143 122 58 46 94 100 58 10 124 137 112 75 41 48 39 18
    </Values>
</Descriptor>
```

Fig. 4: Sample of Colour Structure Description in XML format

3.1.3 Scalable Colour Descriptor

This descriptor uses HSV colour space and uniform quantisation to 256 bins (16 levels in H, 4 in S, and 4 in V). In order to lower the 1024 bit/histogram representation, histograms are encoded using a Haar Transform.

The resulting scaling can be 256, 128, 64, 32, or 16. Fig. 5 shows a sample of Scalable Colour Description in XML format.

```
<Descriptor xsi:type="ScalableColorType" NumberOfCoefficients="64"
                    NumberOfBitplanesDiscarded="3">
    <Coefficients>9 0 6 0 -2 0 -2 0 1 2 -3 -1 -4 0 -1 1 0 0 0 1
                  -1 0 0 0 -1 0 -1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
                  1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
    </Coefficients>
</Descriptor>
```

Fig. 5: Sample of Scalable Colour Description in XML format

3.1.4 Edge Histogram Descriptor

Firstly, the image is spatially decomposed into 16 sub-images using a fixed grid with equal-size rectangles. Then, five masks are used to assign an edge category to a number of sub-blocks in which the sub-images are divided. It is an aggregation of sixteen 5-bin histograms, which captures the spatial distribution of directional edges within each sub-image. The defined types of edges are: horizontal, vertical, 45°, 135°, and non-directional edges. Fig. 6 shows a sample of Edge Histogram Description in XML format.

```
<Descriptor xsi:type="EdgeHistogramType">
    <BinCounts>0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
                 2 6 2 2 1 3 5 3 5 2 3 5 3 4 3 3 5 4 4 4
                 5 3 4 5 4 5 2 4 4 1 6 1 4 5 2 4 3 5 5 3
                 2 5 3 2 5 2 3 3 4 4 0 3 2 2 2 0 3 2 1 2
    </BinCounts>
</Descriptor>
```

Fig. 6: Sample of Edge Histogram Description in XML format

3.1.5 Homogeneous Texture Descriptor

This descriptor extracts values from a frequency layout to give a quantitative characterization of the image texture in terms of directionality, coarseness (granularity), and regularity of patterns. It uses a bank of scale and orientation sensitive filters to estimate the energy and the energy deviation. Filters are applied on individual feature channels. Feature channels are filtered applying 2-D-Gabor functions, which are modulated Gaussians. Fig. 7 shows a sample of Homogeneous Texture Description in XML format.

```
<Descriptor xsi:type="HomogeneousTextureType">
    <Average>184</Average>
    <StandardDeviation>102</StandardDeviation>
    <Energy>224 220 205 167 209 226 196 168 172 160
            170 162 172 148 130 164 139 147 142 114
            105 136 121 106 139 91 85 112 95 87
    </Energy>
    <EnergyDeviation>223 224 206 161 212 229 186 164
            172 157 163 153 164 142 113 152 128 141
            138 95 99 119 115 101 142 78 67 106 71 76
    </EnergyDeviation>
</Descriptor>
```

Fig. 7: Sample of Homogeneous Texture Description in XML format

3.2 Selection of Descriptor Elements

Selection of descriptor elements becomes an important issue in defining the structure of feature vectors. Tab. 2 shows the different configurations of colour and textures components in the feature space. Configuration providing the best balance is indicated in the third column. Balance between descriptor elements is computed by

$$\text{balance}(\mathbf{d}_{colour}, \mathbf{d}_{texture}) = \arg min(\ \text{abs}\ (\ \|\mathbf{d}_{colour}\| - \|\mathbf{d}_{texture}\|\ )\ ) \tag{3}$$

where $\|\mathbf{d}_{colour}\|$ and $\|\mathbf{d}_{texture}\|$ are the number of constituent elements used by colour and texture descriptors, respectively.

Tab. 2: Setting feature vector dimension according to different configurations of colour and texture components in the feature space

Descriptor	Number of elements					Settings
CLD	12					12
CSD	32	64	120	184		64
SCD	16	32	64	128	256	64
EHD	80					80
HTD	32	62				62
$\|\mathbf{d}_{colour}\| = 140,\ \|\mathbf{d}_{texture}\| = 142,$						
					Balance	2
				Feature vector dimension		282

As indicated in column *Settings*, CLD is set up to the default recommended that includes six Y coefficients and three each of Cr and Cb coefficients [13]. CSD is adjusted to 64 elements, which are calculated based on approximations computed using the 184-bin descriptor. SCD is fixed to 64 by scaling the quantised representation of Haar coefficients to obtain the desired number of bits. EHD uses its default corresponding to 80 elements. HTD uses the full layer, it is to say 62 elements.

The 140-element colour and 142-element texture components produce a difference of two elements, which is an acceptable balance. Consequently, the full-size feature vector consists of 282 descriptor elements.

4 EXPERIMENTAL STUDIES

Image features work with different levels of effectiveness depending on the characteristics of the specific image data set. Quality fo the proposed feature vector structure is evaluated applying clustering analysis under different test conditions. The experimental material consists of 1,000 colour images of different size collected from Corel stock gallery and downloaded from FreeFoto.com (ref [www.freefoto.com]). Images are manually categorized into five classes namely *animal*, *building*, *city view*, *landscape*, and *vegetation*.

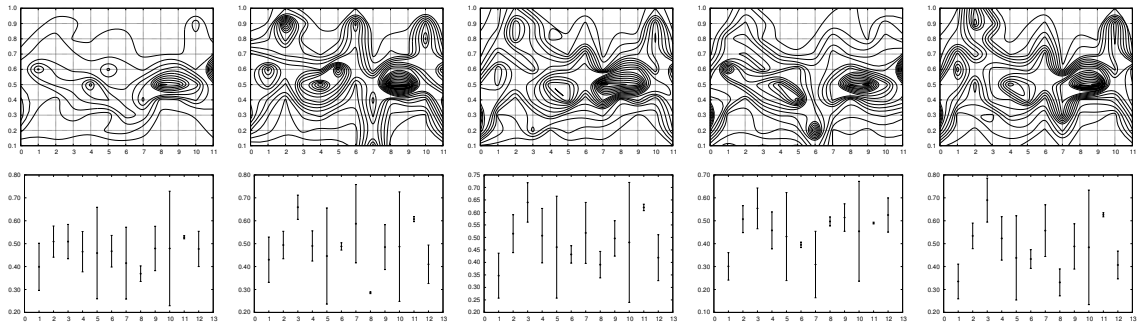
Each class comprises 200 images. The training and testing datasets used for classification is randomly generated; Percentage of training samples is ranged from 10% to 60% of all images and the remaining is allocated for testing.

4.1 Variability of Image Descriptions

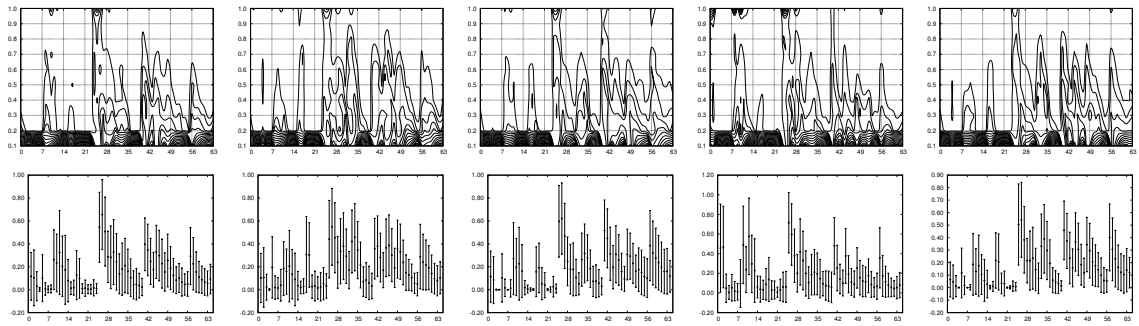
It is worth to get an insight of the underlying structure of data before moving into analysing inter-class separability. The learning space is divided into 5 sub-spaces, each formed with elements from specific visual descriptors. Statistical moments (mean and standard deviation) are used to describe the distribution of the image within each class. They provide information on the location and dispersion of feature values (univariate analysis). In addition, contour plots are used to facilitate identification of feature values concentrated within certain intervals.

Vertical and horizontal axes in the contour plot correspond to the independent variables. X axis indicates the index of the descriptor element associated to the feature values. On the other hand, y axis has been divided into ten intervals to identify the approximate value of the iso-responses (contour lines). High frequency of data points within certain intervals can be observed when iso-responses look like “fingerprints”. Distinguishing features report their frequent data points at different locations along the y axis.

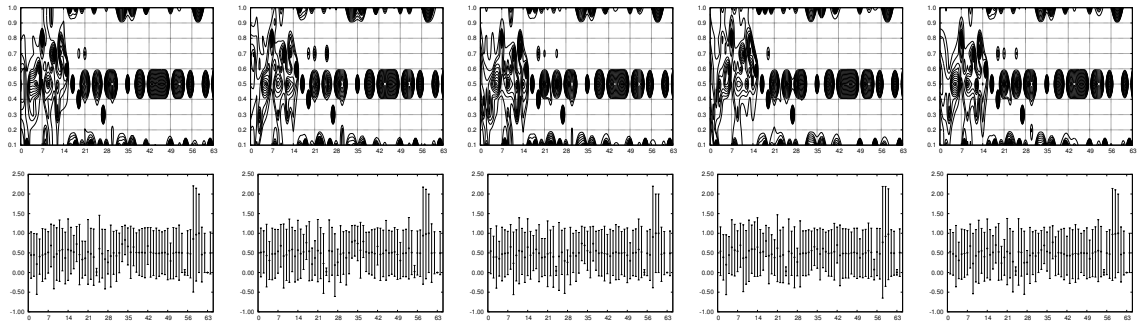
Fig. 8(a) to Fig. 9(b) show contour plots (left column) and mean values and standard deviations (right column) of colour and texture descriptors. Co-ordinates of each descriptor element are placed along the x axis. Contours indicate frequency of feature values in the corresponding intervals (y axis). Original feature values were transformed linearly using min-max normalization.



(a) Colour Layout Descriptions



(b) Colour Structure Descriptions

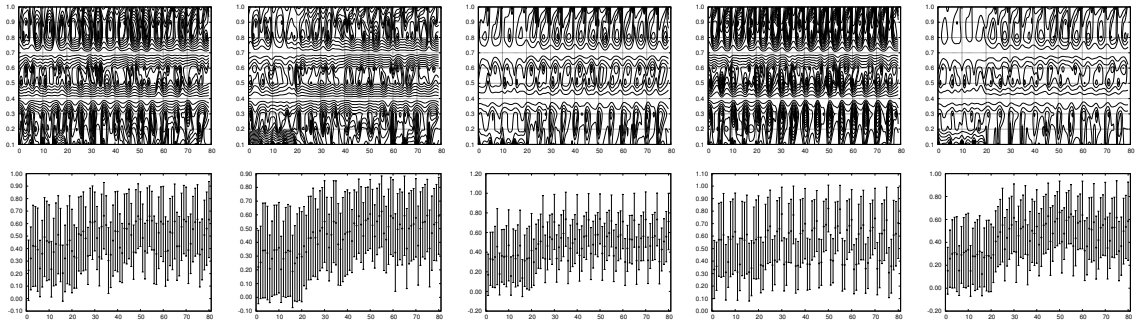


(c) Scalable Colour Descriptions

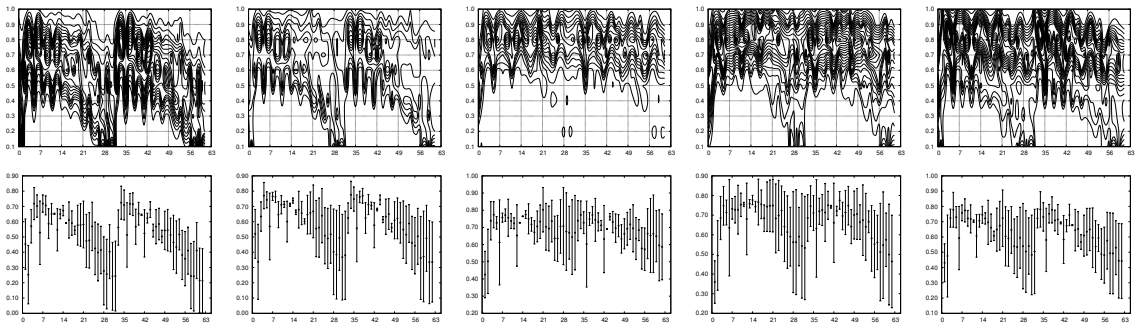
Fig. 8: Contour plots (odd rows), mean values and standard deviations (even rows) of colour descriptions. Descriptor element indexes are placed along the x-axis. Contours indicate frequency of values in the corresponding intervals (y-axis). Classes *animal*, *building*, *city view*, *landscape*, and *vegetation* are arrayed by column from left to right.

In general, there is a high concentration of values around certain intervals in all components. It could derive onto overlapping of classes and low differentiation between features among them. On the other hand, high variability reduces discrimination capabilities at feature level. A salient feature is not a strong representative of its class when there is a high variability intra-class. Overlapping and weak salient features have undesirable effects on classification outcomes.

It stresses the need of a suitable vector structure to get some discriminative properties between image representations. Fig. 10 shows the improvement on classification accuracy when using combined description elements as feature vectors.



(a) Edge Histogram Descriptions

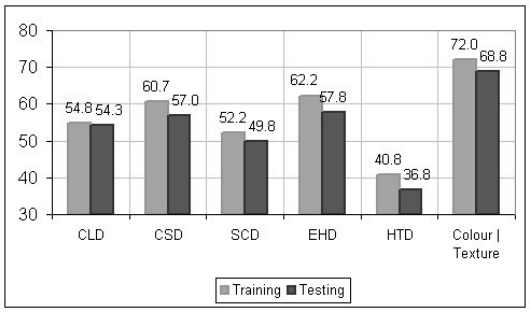


(b) Homogeneous Texture Descriptions

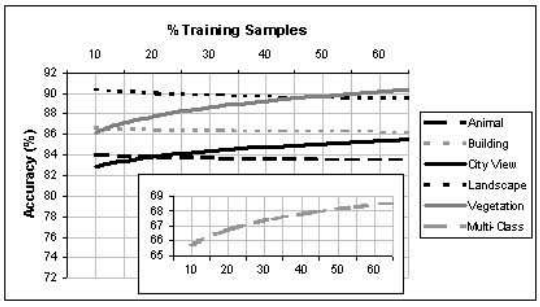
Fig. 9: Contour plots (odd rows), mean values and standard deviations (even rows) of texture descriptions. Descriptor element indexes are placed along the x-axis. Contours indicate frequency of values in the corresponding intervals (y-axis). Classes *animal*, *building*, *city view*, *landscape*, and *vegetation* are arrayed by column from left to right.

4.2 Clustering Analysis

Clustering analysis is used to evaluate semantic grouping of the proposed vector structure. Cluster centres (prototypes) are used to classify new data points. Fig. 10 presents accuracies achieved by the two-class (e.g. animal vs. not animal) and multi-class classifier. It also shows the classification rate obtained when using descriptors either in a separated or in a combined way. Classification accuracy is computed using confusion matrices.



(a) Different Description Elements



(b) Different Training Datasizes

Fig. 10: Classification rates under different test conditions. (a) Accuracy achieved using single-descriptor and multi-descriptor elements (b) Accuracy of two-class and multi-class classifiers using colour and texture descriptions

Similarity of best-ranked images in the clusters, based on their probability of membership to certain class,

resembles partially the expected semantic grouping. Fig. 11 presents samples of images correctly classified. Fig. 12 shows groups with partial misclassification, it is to say groups in which most of the images are ascribed to the same class. Fig. 13 depicts samples of misclassified images. As can be observed, there is a low-level matching (e.g. similar colours) though the semantic grouping is not satisfied. Fig. 14 shows some outlier images whose low-level features are dissimilar to the rest of the images and consequently appear isolated in clusters with few elements (one or two vectors).

Fig. 11: Samples of images correctly classified. First row: City View, second row: Animal, and third row: Vegetation. Probability is indicated below each image

Fig. 12: Samples of classification outcome containing few misclassified images. First row: Landscape (4th image: Animal) and second row: Building (1st and 2nd images: City View). Probability is indicated below each image

5 CONCLUSIONS

A feature vector structure for building an MPEG-7 learning space was proposed. It combines visual descriptors keeping discriminative elements and topological information. Resulting feature space is used for image classification.

Feature vectors use descriptor elements without any transformation, which preserves the semantics embedded in the original descriptions. As illustrated in the clustering outcomes, low-level matching is in certain grade visually detected due to the closeness of the selected descriptors to the human perception.

Although there is some overlapping in the MPEG-7 visual descriptors, the proposed aggregation reduces the description overriding. Chosen settings for each descriptor satisfy the desired balance between components whilst using a minimum number of required descriptor elements to control the vector dimensionality.

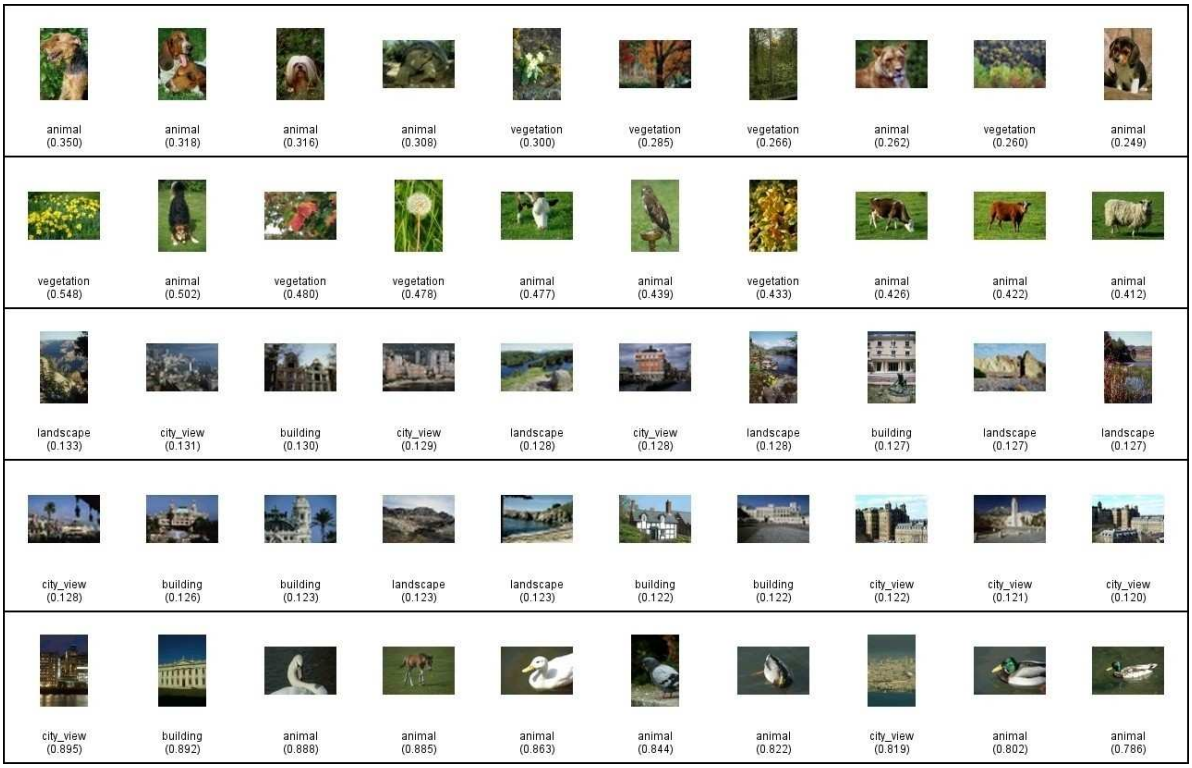


Fig. 13: Samples of misclassified images. Classification outcome presents images from different categories satisfying low-level matching. Probability is indicated below each image

Fig. 14: Samples of outlier images

Acknowledgment

Support from the Natural Sciences and Engineering Research Council (NSERC) and Canada Research Chair (W. Pedrycz) is gratefully acknowledged. Part of the research leading to this paper was also done within the framework of the Network of Excellence SCHEMA.

References

[1] S.-F. Chang, T. Sikora, and A. Puri. Overview of the MPEG-7 standard. *IEEE Trans. Circuits and Systems for Video Technology*, 11(6):688–695, Jun 2001.

[2] A. Dorado and E. Izquierdo. Fuzzy color signatures. In *Proc. IEEE Int'l Conf. on Image Processing*, volume 1, pages 433–436, Rochester, New York, USA, Sep 2002.

[3] R.O. Duda, P.E. Hart, , and D.G. Stork. *Pattern Classification.* J. Wiley, New York, NY, USA, 2nd edition, 2001.

[4] H. Eidenberger. How good are the visual MPEG-7 features? In *Proc. Visual Communications and Image Processing*, volume 5150 of *Proc. of SPIE*, pages 476–488, 2003.

[5] H. Eidenberger. Statistical analysis of content-based mpeg-7 descriptors for image retrieval. *Multimedia Systems*, 10(2):84–97, Aug 2004.

[6] I.K. Fodor. A survey of dimension reduction techniques (ucrl-id-148494). Technical report, Center for Applied Scientific Computing Lawrence Livermore National Laboratory, Livermore, CA, Jun 2002.

[7] U. Guntzer, W.-T. Balke, and W. Kiebling. Optimizing multi-feature queries for image databases. In *Proc. Conf. on Very Large Databases*, Aug 2000.

[8] J.C. Harsanyi and C.-I. Chang. Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach. *IEEE Trans. Geoscience and Remote Sensing*, 32(4):779–785, Jul 1994.

[9] E. Izquierdo, I. Damnjanovic, P. Villegas, X. Li-Qun, and S. Herrmann. Bringing user satisfaction to media access: The 1st busman project. In *Proc. 8th IEEE Int'l Conf. on Information Visualisation*, pages 444–449, London, UK, Jul 2004.

[10] M. Kokare, B.N. Chatterji, and P.K. Biswas. Dimensionality reduction of tree structured wavelet transform texture features for content based image retrieval. In *Proc. 7th IEEE Int'l Conf. on Control, Automation, Robotics, and Vision*, volume 3, pages 1647–1625, Dec 2002.

[11] J. Laaksonen, M. Koskela, and E. Oja. PicSOM -self-organizing image retrieval with MPEG-7 content descriptors. *IEEE Trans. Neural Networks: Special Issue on Intelligent Multimedia Processing*, 13(4):841–853, Jul 2002.

[12] B. S. Manjunath, P. Salembier, and T. Sikora. *Introduction to MPEG-7 Multimedia Content Description Interface*. J. Wiley, New York, 2002.

[13] B.S. Manjunath, J.-R. Ohm, V.V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Trans. Circuits and Systems for Video Technology*, 11(6):703–715, Jun 2001.

[14] V. Mezaris, H. Doulaverakis, R. Medina Beltran de Otalora, S. Herrmann, I. Kompatsiaris, and M. G. Strintzis. *Image and Video Retrieval: Third International Conference, CIVR 2004 Proceedings*, volume 3115 of *Lecture Notes in Computer Science*, chapter A Test-Bed for Region-Based Image Retrieval Using Multiple Segmentation Algorithms and the MPEG-7 eXperimentation Model: The Schema Reference System, pages 592–600. Springer-Verlag GmbH, Dublin, Ireland, Jul 2004.

[15] T. Ojala, M. Aittola, and E. Matinmikko. Empirical evaluation of MPEG-7 XM color descriptors in content-based retrieval of semantic image categories. In *Proc. 16th Int'l Conf. on Pattern Recognition*, volume 1, pages 701–706, 2002.

[16] T. Ojala, T. Mäenpää, J. Viertola, J. Kyllönen, and M. Pietikäinen. Empirical evaluation of MPEG-7 texture descriptors with a large-scale experiment. In *Proc. 2nd Workshop on Texture Analysis and Synthesis*, pages 99–102, 2002.

[17] J. R. Smith, S. Basu, C.-Y. Lin, M. Naphade, and B. Tseng. Integrating features, models, and semantics for content-based retrieval. In *Proc. NSF Workshop in Multimedia Content-Based Indexing and Retrieval*, Sep 2001.

[18] J. R. Smith and B. Lugeon. A visual annotation tool for multimedia content description. In *Proc. SPIE Photonics East, Internet Multimedia Management Systems*, Nov 2000.

[19] R. Weber, H.-J. Shek, and S. Blott. A quantitative analysis and performance study for similarity search methods in high-dimensional spaces. In *Proc. Conf. on Very Large Databases*, New York, NY, USA, 1998.

[20] P. Wu, B.S. Manjunath, and H.D. Shin. Dimensionality reduction for image retrieval. In *Proc. IEEE Int'l Conf. on Image Processing*, volume 3, pages 1647–1652, Sep 2000.

Un algoritmo ACO para la compresión de imágenes[1]

Cristian A. Martinez
Universidad de Buenos Aires, Departamento de Computación, Fac. de Cs. Exactas y Naturales
Buenos Aires, Argentina, C1428EGA
Email: martinezdro@{argentina.com, yahoo.com.ar}

Abstract

This work is an application of the Ant Colony Metaheuristic (ACO) to the problem of fractal compression of images using IFS.
An ACO hybrid algorithm is proposed for the fractal compression of images and the results obtained are shown. According to the tests carried out, the algorithm proposed offers images with similar quality to it obtained by a deterministic method, but in an approximate time of 37% of the previous one.

Keywords: Algorithms, Metaheuristics

Resumen

Este trabajo es una aplicación de la metaheurística Colonia de Hormigas (ACO) al problema de la compresión fractal de imágenes usando IFS.
Se propone un algoritmo híbrido ACO para la compresión fractal de imágenes y se muestran los resultados obtenidos en diferentes tests. De acuerdo a las pruebas realizadas, el algoritmo propuesto ofrece imágenes con calidad similar a la obtenida por un método determinístico, pero en un tiempo aproximado al 37% del anterior.

Palabras claves: Algoritmos, Metaheurísticas

1. Introducción

Desde hace más de 15 años, el campo de la compresión de datos ha tenido un gran auge, causado por el gran volumen de información existente (documentos, videos, audio, imágenes) personal y profesional que manejamos cotidianamente. Constantemente, usamos software de compresión de datos para almacenar o transmitir información.
Las técnicas de compresión de datos intentan disminuir la redundancia en el contenido de un archivo, logrando archivos más livianos en cuanto a espacio físico.
Dentro de la compresión de datos, surge la compresión fractal de imágenes[2]. Una definición de fractal extraída de [11], dice que "...es una imagen que puede ser completamente descripta por un algoritmo matemático en su más fina textura y detalle...". Podemos decir, que la compresión fractal consiste en la obtención de una aproximación de la imagen real, a través de un conjunto de transformaciones aplicadas sobre ciertos bloques de la imagen. Una limitante de estas técnicas es el tiempo computacional necesario para comprimir imágenes.
Una analogía con la conducta de las hormigas reales fue presentada como un nuevo paradigma llamado ACO (Ant Colony Optimization). Las principales características que posee son la rápida búsqueda de buenas soluciones, el trabajo en paralelo y el uso de información heurística, entre otros. Existen varios problemas resueltos con ACO: TSP, Knapsack, Cutting Stock, Graph Coloring, Job Shop, etc.
El trabajo propone un algoritmo híbrido ACO para la compresión fractal de imágenes, problema aún no resuelto con ACO.
En la sección 2, se describen aspectos teóricos de compresión fractal y ACO. En la sección 3, se detalla el algoritmo propuesto y las variaciones respecto de los algoritmos conocidos de ACO. En la sección 4, se muestran resultados de diferentes tests realizados aplicando variaciones en los parámetros y comparándolos con un algoritmo determinístico. Por último, en la sección 5, se mencionan aspectos a tener en cuenta en la implementación de algoritmos ACO basados en la experiencia alcanzada durante la implementación, como así también posibles mejoras al algoritmo presentado.

[1]Parcialmente financiado por el Proyecto PICT 11-09112 de la Agencia de Promoción Científica
[2] El término imagen debe asociarse a imágenes en escala de grises.

2. Compresión Fractal y Colonia de Hormigas

2.1. Compresión Fractal

Almacenar la imagen de un helecho como una colección de píxels requiere mucho espacio en memoria si se exige buena resolución, supongamos 256 kb. (para una imagen de 256*256). Sin embargo, es posible almacenar una colección de números cercanos a 64 kb., que definen ciertas transformaciones tal que son capaces de generar la imagen del helecho cuando se desee. Ahora, supongamos que disponemos de cualquier imagen. Si una colección de transformaciones puede generar de nuevo la imagen, entonces podría almacenarse dicha colección de manera compacta. La compresión fractal describe el esquema para obtener dicha colección. Para más detalles, ver [7].

Supongamos que tenemos una imagen I que deseamos comprimir. Esto significa que buscamos una colección de transformaciones $w_1, w_2, ...w_n$ siendo $W = \bigcup w_i$, de manera tal de que $I = |W|$, es decir

$$I = W(I) = \bigcup w_i(I) \quad \forall i = 1..n$$

Para lograr esto, debe particionarse I en bloques tal que al aplicar las transformaciones w_i se obtenga la imagen I original. Sin embargo, existen imágenes que no se componen de bloques que puedan ser transformadas fácilmente y obtener exactamente I. Pero podemos obtener $I' = |W|$ siendo la distancia d(I,I') lo más pequeño posible. Entonces lo que se buscará es minimizar la distancia entre los bloques de la imagen con los bloques transformados, o sea:

$$Min\ d(I \bigcap (R_i x I), w_i(I))\ i = 1....n$$

Esto es, debemos buscar bloques de dominio D_i y transformaciones w_i , tal que al aplicar la transformación sobre el bloque de dominio estemos lo más cerca posible de un bloque de la imagen R_i (o región).

Algunas definiciones:

- Una transformación w en el espacio n-dimensional R^n es una función de R^n en R^n . Una transformación afín $w : R^n \to R^n$ puede ser escrita como $w(x) = Ax + b = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & .... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{pmatrix}$ siendo A una matriz de nxn en R^{n*n} llamada matriz deformación[3] y el vector b en R^n llamado vector traslación.
- Una transformación afín w o una matriz de deformación A, es contractiva si $\|A\| < 1$. Se dice contractiva con factor s, para algún $0 \le s \le 1$, si $\|A\| < s$.
- Sea un espacio vectorial R^n y una métrica d, una sistema de función iterada IFS es un conjunto finito de transformaciones contractivas $W = \{w_1, w_2, ...w_m\}$. El factor de contractividad s se define como el máximo de los factores de contractividad de las transformaciones $s = \max\{\|w_1\|, \|w_2\|, ...\|w_m\|\}$. Dado un IFS, define su transformación asociada sobre el espacio de subconjuntos compactos, $D(R^n)$ como $W(I) = \bigcup w_i(I)$ para todo $I \in D(R^n)$.
- Dado un IFS $\{w_1, w_2, ..w_m\}$ existe un único conjunto invariante I tal que $I = \bigcup_{i=1}^{m} w_i(I)$. I se conoce como el atractor del sistema.
- Dada una métrica d sobre el espacio n-dimensional R^n , dos conjuntos $A, B \in R^n$, se define distancia entre los dos conjuntos como $d(A,B) = \max\{\sup_{x \in A}\{\inf_{y \in B} d(x,y)\}, \sup_{y \in B}\{\inf_{x \in A} d(x,y)\}\}$. Esta función distancia es conocida como métrica Hausdorff.
- Para saber cuánto aproxima el atractor de un IFS a una imagen dada, debemos definir (y conocer) el Teorema del Collage. Sea $W = \{w_1, w_2, ...w_m\}$ un IFS con factor de contractividad s. Sea $W : D(R^n) \to D(R^n)$ su transformación asociada y sea $A \in D(R^n)$ el atractor de W. Entonces

[3] En el espacio bidimensional, la matriz deformación puede ser descompuesta en cuatro pasos: escalar, rotar, torcer y estirar.

$d(L,A) \le \frac{d(L,W(L))}{1-s}$ para cualquier $L \in D(R^n)$. Por lo tanto, el problema de encontrar un atractor A cercano a una imagen I, es equivalente a minimizar la distancia $d(I, \bigcup_{i=1}^{m} w_i(I))$.

2.2. Algoritmo determinístico de Compresión Fractal

El siguiente algoritmo fue propuesto por Barnsley [1]. Propone dividir la imagen en bloques no superpuestos de rango $R = \{R_i\}$ y bloques de dominio $D = \{D_i\}$ en el que puede existir solapamiento parcial. Para cada bloque de rango R_i se realiza una búsqueda exhaustiva sobre los bloques de dominio, hasta encontrar el mejor bloque D_j tal que la distancia $d(R_i, wi_{jk}(D_j))$ sea la menor posible. Entonces, se almacenan las coordenadas del bloque de dominio D_j y el tipo de transformación aplicada al mismo.

```
For i=0 to #img-1
        min= α
        For j=0 to #dom-1
                For k=0 to #transf-1
                        Dist=distance(R[i],w(k,D[j]))
                        If Dist<min then
                                Min=Dist
                                Dj=j
                                Wi=k
        Save (dj,wi)
```

Fig. 2.2.1 - Algoritmo Exhaustivo

Una característica de la compresión fractal es que la resolución de la descompresión es independiente del tamaño de la imagen, lo que significa que el tamaño elegido para la descompresión puede ser diferente al de la imagen original, sin que esto altere la resolución.

2.3. ACO

Las colonias de hormigas son sistemas distribuidos que forman organizaciones sociales altamente estructuradas, lo que permite que puedan realizar tareas complejas que en ciertos casos, exceden las capacidades individuales de un simple agente ([2]).

Básicamente, las hormigas salen desde el hormiguero para buscar alimento dejando un rastro en el suelo, llamado feromona. Si el camino que eligió alguna hormiga fue más corto en comparación con el tiempo de otras, podrá depositar el alimento en el hormiguero y regresar nuevamente a buscar alimento y de esta forma, incrementar el volumen de feromona en el camino previamente elegido, provocando que las demás hormigas sigan el mismo camino basado en el alto nivel de feromona dejado en el suelo.

Basado en la observación de las conductas reales de las hormigas, surgieron los algoritmos ACO que son usados para resolver problemas de optimización.

Uno de los algoritmos más conocidos de ACO, es el Sistema de Hormigas (AS). Fue creado por Marco Dorigo y la primera aplicación conocida fue en 1992, en el que resolvió un problema de TSP, basado justamente, en el hecho de que las hormigas encuentran el camino más corto para ir desde el hormiguero hasta la fuente de alimento.

2.3.1 AS-ACO

Mencionaremos algunos detalles del algoritmo AS-ACO que adapta la conducta real de las hormigas para alcanzar el costo mínimo en problemas de grafos.

Asociamos a cada arista (i,j) del grafo G=(N,A) una variable τ_{ij} llamada rastro artificial de feromona. Como se mencionó anteriormente, las hormigas siguen los rastros de feromona. La cantidad de feromona depositada en cada arista es proporcional a la utilidad lograda.

Cada hormiga construye desde el nodo inicial, paso a paso, una solución al problema. En cada nodo, la hormiga usa la información local almacenada para decidir a qué nodo moverse.

Cuando una hormiga k, se encuentra en un nodo i, usa los rastros de feromona τ_{ij} para obtener la probabilidad de elegir j como nodo próximo:

$$p_{ij}^{\ k} = \begin{cases} \dfrac{\tau_{ij}^{\ \alpha}}{\sum \tau_{il}^{\ \alpha}} & si\ \ j \in \mathrm{N}_i^{\ k} \\ 0 & si\ \ j \notin \mathrm{N}_i^{\ k} \end{cases} \qquad (1)$$

siendo $\mathrm{N}_i^{\ k}$ la vecindad de la hormiga k, estando en el nodo i. La vecindad de un nodo i son todos los nodos conectados directamente en el grafo G=(N,A), menos su predecesor, evitando de esta forma, volver sobre nodos visitados previamente.
Eventualmente, la hormiga k llegará al nodo final, aplicando nodo a nodo la política de decisión, indicada en la ecuación 1.
Una vez que la hormiga k completó su camino, deposita una cantidad de feromona sobre las aristas que visitó, dada por la siguiente ecuación:

$$\tau_{ij} = \tau_{ij} + \Delta\tau^k \qquad (2)$$

de esta forma, cuando las próximas hormigas estén en el nodo i, habrá grandes chances de que elijan al nodo j como próximo, debido a la concentración de feromona. Un aspecto importante es el valor que adopta $\Delta\tau^k$. En general, la cantidad de feromona depositada es una función no creciente de la longitud del camino.
La evaporación del rastro de feromona puede entenderse como un mecanismo que permite la rápida convergencia de las hormigas hacia un camino subóptimo (ver [4]). De hecho, decrementar la cantidad de feromona favorece a la exploración de diferentes caminos durante el proceso de búsqueda y evitar caer en óptimos locales. En las colonias de hormigas reales, la evaporación ocurre, pero no juega un rol importante en la búsqueda del camino más corto.
La evaporación favorece al olvido de errores o malas decisiones en el pasado, permitiendo una continua mejora en las soluciones alcanzadas.
La evaporación del rastro de feromona es aplicada a través de la siguiente ecuación

$$\tau_{ij} = (1-\rho)\tau_{ij} \qquad (3)$$

para toda arista (i,j) perteneciente al grafo G y siendo $\rho \in (0,1]$ un parámetro.
Un esquema de alto nivel de Colonia de Hormigas es:

```
Procedure Colonia__hormigas
    Initialize
    While not terminate
            Construct_solution
            Apply_localsearch
            Update_pheromone
```

Fig. 2.3.1.1 – Algoritmo AS-ACO

3. Detalles de Implementación

Esta sección explica como la metaheurística ACO se adaptó al problema de compresión de imágenes. Además, se detallan ciertos aspectos que tienen que ver con la compresión propiamente dicha.

3.1. Feromona

La calidad de una aplicación ACO depende en cierta medida de la definición del rastro de feromona. La definición de la misma depende del problema a resolver. Por ejemplo en el problema del TSP de [3], la feromona que se deposita en el par (i,j) denota la conveniencia de ir a la ciudad j, desde la ciudad i. En [10] donde se resuelve el problema de Bin Packing, el rastro de feromona depositada en el par (i,j) refleja el beneficio de tener los ítems i y j en el mismo Bin. En este trabajo, la feromona en el par (i,j) relaciona al bloque de rango i y al bloque de dominio j. La matriz de feromona es rectangular (no simétrica) donde las filas representan a los bloques de rango (bloque de la imagen) y las columnas los de dominio (bloques de referencia para transformar).

3.2. Información Heurística

La posibilidad de usar información heurística para dirigir la construcción probabilística de las soluciones realizada por las hormigas, es importante porque permite explotar el conocimiento del problema. Este conocimiento puede

estar disponible a priori (problemas estáticos) o en tiempo de ejecución (problemas dinámicos). En los problemas estáticos, la información heurística η es computada en cada corrida del algoritmo. Ejemplos de esta situación son la inversa de la distancia entre las ciudades i y j usado en el problema de TSP de [5] y el tamaño del ítem j usado en el problema de Bin Packing de [10]. La información heurística estática tiene la ventaja de ser computada en cada iteración del algoritmo y puede ser calculado conjuntamente con la información de feromona. En el caso de problemas dinámicos, la información heurística depende de la solución parcial construida y debe ser calculada en cada paso realizado por las hormigas.
Debe tenerse en cuenta que el uso de la información heurística es importante para un algoritmo ACO y su importancia decrece si se usan algoritmos de búsqueda local. Según el problema a resolver, será mas o menos compleja definir la información heurística. En la sección 4, se analiza la calidad de las soluciones devueltas del algoritmo usando información heurística y búsqueda local.
En nuestro trabajo, se probaron varias alternativas de información heurística, siendo la más favorable la *inversa del error cometido al utilizar el bloque de dominio j para aproximar el bloque rango i.* Es dinámica en cuanto a su obtención, a diferencia de otros problemas resueltos con ACO donde la información es estática (TSP, Bin Packing, Snapsack, Job Shop, etc); esto es así, ya que no se dispone del error de aproximar un bloque de dominio, antes del proceso de optimización y por lo tanto es vital la exploración exhaustiva del espacio de soluciones. Se presentarán más detalles sobre las otras opciones, en la sección 5.

3.3. Construcción de Soluciones
Como se indicó anteriormente, las hormigas construyen soluciones factibles usando información heurística y feromona. Cada hormiga construye su camino eligiendo para cada bloque de rango i, un bloque de dominio j. La probabilidad que una hormiga k elija un bloque de dominio j para el bloque rango i, está dada por la siguiente ecuación:

$$p_{ij}^{\,k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum \tau_{il}^{\alpha}\eta_{il}^{\beta}} & si\ q < e^{-\frac{k}{2*ants}} \\ J & coc \end{cases} \qquad (4)$$

donde

- $\eta_{ij} = \dfrac{1}{err_{ij}}$, y err_{ij} es el error cometido al seleccionar el bloque j para aproximar al bloque i.
- q es una variable aleatoria uniforme [0,1). A diferencia de [4], donde muestra que Ant Colony System (ACS), usa un parámetro fijo q_0 que sirve de umbral para la selección de un cálculo de probabilidad o una variable, en este trabajo se consideró la característica de Simulated Annealing, en el hecho de elegir malas soluciones a través de una comparación con una expresión exponencial. Además, hay que denotar que las hormigas usadas tienen diferentes clases. Básicamente, lo que se hizo fue utilizar hormigas buenas y malas para la construcción de soluciones. La expresión en sí, permite elegir bloques "malos" al comienzo del proceso de búsqueda de soluciones y se basa en la cualidad de Colonia de Hormigas de no precisar buenas soluciones para su proceso, lo que permitió aprovechar esa instancia para la exploración de nuevos espacios de soluciones. Una aplicación de Simulated Annealing puede encontrarse en [12].
- J es una variable aleatoria que selecciona un bloque "malo" del entorno de i.

La construcción de una solución, entonces, consiste en elegir según el resultado de la ecuación anterior, un bloque de dominio j para cada bloque de rango i.

3.4. Actualización de Feromona
Luego que todas las hormigas construyeron sus respectivos caminos, la información de feromona es actualizada aplicando evaporación, seguido por el depósito de nueva feromona como se indica a continuación:

$$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}^{\sup er} \qquad (5)$$

donde $\Delta\tau_{ij}^{\sup er} = 1/C^{\sup er}$, siendo $C^{\sup er}$ el valor objetivo alcanzado por la "súper" hormiga. El criterio seguido es similar a MAX-MIN ([4]), donde sólo una hormiga es la que deposita feromona en cada iteración. La versión de MAX-MIN es sencilla de implementar y ofrece buenos resultados. Sin embargo, en este trabajo, se genera por cada iteración una "súper" hormiga que construye un camino formado por los menores errores cometidos por las otras

hormigas. Es decir, las hormigas realizan trabajo cooperativo para que sólo una hormiga deposite feromona en el mejor camino formado por las mejores selecciones de cada hormiga usada.

3.5. Función Objetivo

Para que el algoritmo propuesto se acerque a buenas soluciones, es necesario contar con una función objetivo que mida la calidad de las soluciones. Así como en [3], la función objetivo era el mínimo costo de ruteo de vehículos, en [5] la mínima distancia de visita a todas las ciudades, en [10] un promedio de uso de bins, se decidió usar una expresión similar al fitness usada en [13] que básicamente mide la distancia entre los bloques de la imagen (rango) y los bloques de dominio.
La expresión a minimizar es:

$$\min \sum_{(x,y)\in r_i} \sum_{j=0}^{\#dom-1} \left[\left(r_i(x,y) - \vec{d}_j(x,y) \right) - \left(\bar{r}_i - \bar{d}_j \right) \right]^2 \quad \forall i = 0..\#img - 1 \qquad (6)$$

donde

$r_i(x,y)$ es el píxel (x,y) del bloque de rango i

$\vec{d}_j(x,y)$ es el píxel (x,y) transformado del bloque de dominio j

$\bar{r}_i$ es la media (en rgb) del bloque de rango i

$\bar{d}_j$ es la media (en rgb) del bloque de dominio j

#dom es el número de bloques de dominio

#img es el número de bloques de rango

3.6. Búsqueda Local y ACO

Los algoritmos ACO mejoran su performance en la búsqueda de buenas soluciones, cuado se usan algoritmos de búsqueda local. Podemos mencionar a [3] donde aplica la heurística 2-opt para la mejora de soluciones en el problema de ruteo, a [10] que usa el criterio de dominancia de Martello y Toth, entre otros.
La razón por la cual los algoritmos ACO con los de búsqueda proveen buenas soluciones radica en el hecho de que son complementarios. Un algoritmo ACO generalmente realiza búsquedas "gruesas", en tanto que los de búsqueda local buscan en la vecindad de la solución, intentando mejorar aún más la calidad de las soluciones.
Respecto a algoritmos de búsqueda local, podemos mencionar al operador de cruzamiento modificado propuesto en el algoritmo genético de [13] y un algoritmo de búsqueda local en [9] que implica demasiado cálculo, donde, tal vez es mejor usar ciertas modificaciones del algoritmo de Jacquin [8], lo que implica un cálculo previo a la búsqueda de soluciones.
Aquí se aplicaron dos algoritmos de búsqueda local.

Algoritmo 1: basados en el hecho de que es posible que existan porciones de la imagen similares, cada bloque de imagen es comparado con los bloques de dominio elegidos por los vecinos.

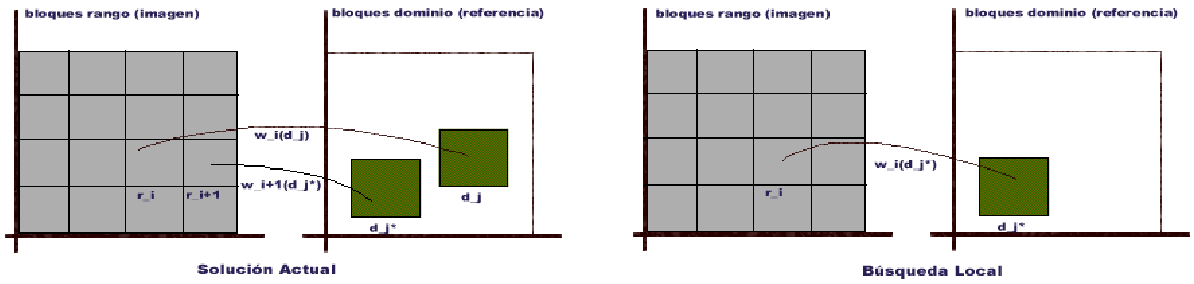


Fig. 3.6.1 – Algoritmo de Búsqueda Local

Algoritmo 2: a fin de mejorar la calidad de las soluciones y explorar nuevos bloques de dominio, para cada bloque rango (de la imagen) cuyo error en la aproximación, supere a la media basada en la mejor solución obtenida hasta el momento, se le asigna aleatoriamente un bloque de dominio aún no usado.

En la siguiente figura, se muestra un pseudocódigo del algoritmo propuesto:

```
Main
        Initialize
        For i=0 to #iters-1
                Construct_solution                      (1)
                For k=0 to #Ants-1
                        Generate_best_path              (2)
                local_search(super)                     (3)
                Update_pheromone(super)                 (4)

(1)  /* cada hormiga construye en forma paralela, su camino basado en la información heurística y la feromona
        depositada, pero permitiendo que algunas hormigas tengan chances de elegir malos bloques */
            for img=0 to #img-1
                for k=0 to #ants-1
                        if u(0,1]< e^(-k/(2#Ants)) then
                                ant[k].dom[img]=best_neighbor[img]
                        else
                                ant[k].dom[img]=no_neighbor[img]

(2)  /* para cada bloque rango de la imagen, se calcula el error y se va construyendo el mejor camino para la súper
        hormiga */
            for img=0 to #img-1
                if error(I,dom(i))<SuperAnt.error(i) then
                        Update(SuperAnt.error(i),error(I, dom(i)))

(3)  /* el orden varía según el número de iteraciones */
            Local_Search1(super)
            Local_Search2(super)

(4) /* se evapora feromona en toda la matriz y se deposita feromona solamente en el camino que posee la súper
            hormiga */
            Evaporate_pheromone
            Deposite_pheromone(super)
```

Fig. 3.6.2 – Pseudocódigo del Algoritmo propuesto

4. Tests

Los resultados de las pruebas realizadas son comentados en esta sección. Primero se definirán los parámetros del algoritmo ACO propuesto y luego se compararán sus resultados con los del algoritmo determinístico propuesto en [11]. Posteriormente, se analizarán resultados logrados por el algoritmo usando búsqueda local y sin la misma.
Los tests fueron realizados sobre varias imágenes de prueba (paisajes, rostros, animales, etc.), obteniéndose resultados similares en calidad y tiempo, a los alcanzados con la imagen de prueba aquí usada.[4]

4.1. Parámetros del Algoritmo Propuesto

Los parámetros que serán definidos corresponden al algoritmo ACO general:

- Número de Hormigas: es el parámetro que indica cuántas hormigas construirán soluciones por cada iteración. Considerando la bibliografía (ver [4]), es recomendable usar 10 hormigas. Sin embargo, se considerarán otros valores que servirán de casos de prueba.
- β: es el parámetro que define la importancia de la información heurística en contraposición a la información de feromona. En [4] se recomienda usar el valor 2, en [10] usa valores 2,5 y 10, mientras que en [3] usa el valor 5. Según nuestros tests, no ofrece mucha relevancia en comparación con otros.

[4] Imagen obtenida del sitio www.sodastereo.com.

- α : es el parámetro que define la importancia del rastro de feromona. En [10] usa 1, en [5] usa 0,1; en tanto en [4] recomienda 1. También presenta la misma característica que β en cuanto a la relevancia en los resultados.
- ρ : es el factor de evaporación. En [6] y [7] usan valores cercanos a 1.
- Criterio de parada: es el número de veces que todas las hormigas en conjunto, construirán nuevas soluciones. Existen otras alternativas para la finalización del algoritmo, como ser la diferencia entre dos soluciones consecutivas, tiempo de cpu, estancamiento, cercanía a la solución óptima, entre otras.
- Factor S: Sirve para obtener una solución inicial. Esta se obtiene a través de la sumatoria del producto entre un factor s y la diferencia de medias entre bloques de rango y dominio.[5]
- Tamaño de bloque: es el parámetro que indica el tamaño de los bloques de rango y dominio. El valor usado es de 8x8.

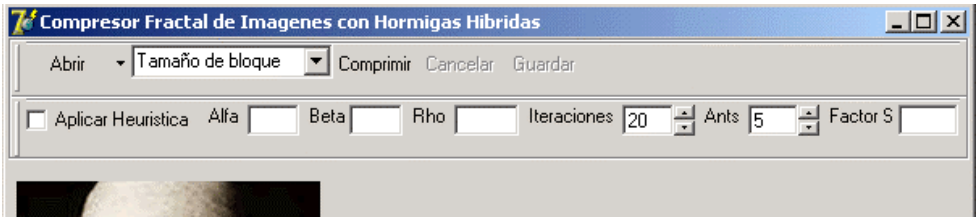


Fig. 4.1.1 – Interfaz del Compresor

4.2. Resultados

Todos los tests fueron realizados en una PC Celeron 750 MHz con 256 Mb Ram, bajo Windows 2000 Server. La aplicación fue desarrollada con Delphi 6/7.

En las siguientes cuatro tablas, se mostrarán resultados obtenidos (tiempo de ejecución y valor objetivo obtenido) por el algoritmo propuesto y la comparación con el tiempo de ejecución y el valor objetivo alcanzado por el algoritmo determinístico.

El resultado óptimo determinístico obtenido para la imagen de prueba es **Valor Función Objetivo,** 4996545[6] y **Tiempo de Ejecución,** 402 segs

Tabla 1: Comparación entre el óptimo determinístico y nuestro algoritmo, con diferentes valores de ρ .

Parámetros fijos usados

α =1, β =2, iteraciones=45, Hormigas=10, Factor S=10,

ρ	**Tpo (Segs)**	**Error (%)**	**Func.**	**Error (%)**
0.2	195	51.5	6196285	24
0.5	195	51.5	6182589	23.7
0.7	192	52.2	6178530	23.7
0.85	203	49.5	6490655	29.9

Según Tabla 1, los mejores resultados corresponden a factores entre 0.5 y 0.7, es decir, se obtiene un buen resultado con un tiempo de ejecución cercano al 50% que el usado por el algoritmo determinístico y está alejado del óptimo determinístico en un 24%. En la siguiente figura, se observa la imagen original, la compresión determinística y el mejor resultado heurístico hasta el momento.

[5] Para la imagen de test, el valor recomendado es 10.

[6] El resultado se obtuvo aplicando la ecuación (6).

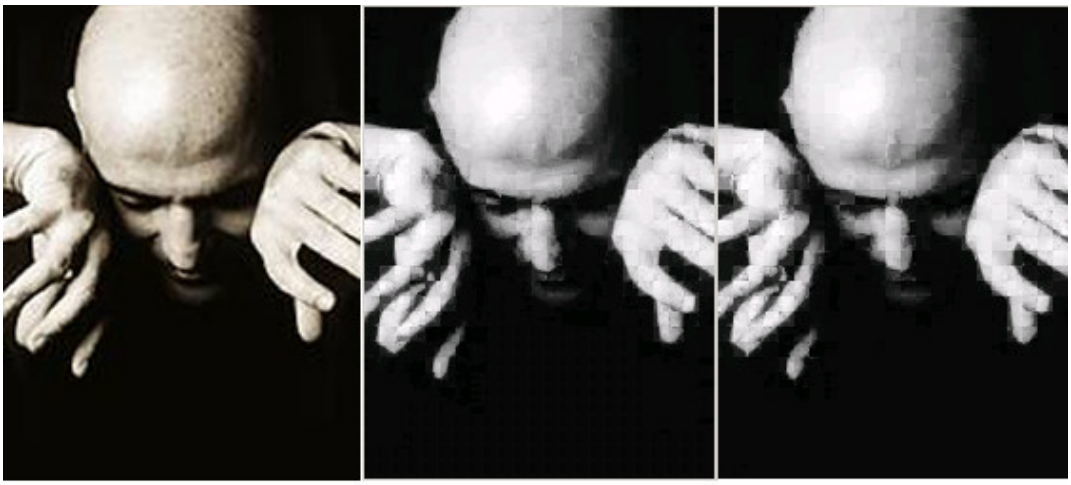


Fig. 4.2.1 - Comparación visual de resultados, entre compresión determinística y heurística

Tabla 2: Comparación entre el óptimo determinístico (**Valor Función Objetivo,** 4996545[7] y **Tiempo de Ejecución,** 402 segs) y nuestro algoritmo, con diferentes valores de α y β .

Parámetros fijos usados
ρ =0.7, iteraciones=45, Hormigas=10, Factor S=10,

α	β	Tpo (Segs)	Error (%)	Func.	Desvío (%)
1	2	192	52.2	6178530	23.7
1	3.5	203	49.5	6298933	26.1
1	5	205	49	6747003	35
0.5	2	205	49	6310030	26.3
0.5	3.5	207	48.5	6090193	21.9
0.5	5	208	48.3	6211330	24.3

Según Tabla 2, los mejores resultados están relacionados con α =0.5, lográndose una solución mejor (21.9% alejada del óptimo) que la alcanzada en Tabla 1. Sin embargo, los tiempos de ejecución son mayores.

Tabla 3: Comparación entre el óptimo determinístico (**Valor Función Objetivo,** 4996545 y **Tiempo de Ejecución,** 402 segs) y nuestro algoritmo, con diferentes valores de hormigas e iteraciones.

Parámetros fijos usados
α =1, β =2, ρ =0.7, Factor S=10

Horms	Iters	Tpo (Segs)	Error (%)	Func.	Error (%)
10	35	161	60	6627192	32.6
10	45	192	52.2	6178530	23.7
10	55	290	27.9	6002001	20.1
15	35	217	46	6281371	25.7
5	45	139	65.4	6274252	25.6
5	55	170	57.7	6171194	23.5

De los resultados obtenidos en la última tabla, podemos decir que:

[7] El resultado se obtuvo aplicando la ecuación (6).

- Con 10 hormigas y 55 iteraciones, es decir, más iteraciones que la mejor solución obtenida hasta la tabla 2, equivale a mejor calidad en la solución pero provoca más tiempo de ejecución (77.1% del tiempo necesario para obtener el óptimo determinístico)
- Usando más hormigas que la mejor solución hasta la tabla 2 y menos iteraciones (15 hormigas y 35 iteraciones) los resultados obtenidos no mejoran a los alcanzados en tabla 2.
- Se obtuvieron mejores tiempos de ejecución y calidad en la solución, que la mejor solución hasta la tabla 2 y corresponden a los resultados de la última fila (42% del tiempo determinístico y un alejamiento del óptimo de un 23.5%). Sin embargo, existe una solución con menor tiempo de ejecución y calidad aceptable (penúltima fila) que será mostrada en la siguiente figura.

Fig. 4.2.2 - Comparación visual de resultados, variando hormigas e iteraciones

Como se observa en la Fig.7, los resultados obtenidos en calidad de imagen para 5 hormigas y 45 iteraciones son "aceptables", logrando un tiempo de ejecución igual al 37% del tiempo logrado por el óptimo determinístico y un alejamiento del óptimo del 25.6%.

Tabla 4: Comparación entre el óptimo determinístico (**Valor Función Objetivo,** 4996545 y **Tiempo de Ejecución,** 402 segs) y nuestro algoritmo, con diferentes cantidades de hormigas e iteraciones, sin búsqueda local.

Parámetros fijos usados
α =1, β =2, ρ =0.7, Factor S=10

Horms	Iters	Tpo (Segs)	Error (%)	Func.	Error (%)
5	55	132	67.2	8984190	79.8
10	45	175	56.5	8179007	63.7
10	55	213	47.0	7838685	56.9

De la Tabla 4, el mejor resultado sin búsqueda local, es mayor al doble de la mejor solución con búsqueda local (23.5%). Además, el tiempo de ejecución es levemente menor.
La siguiente figura muestra la imagen original y las mejores soluciones con y sin búsqueda local.

Fig. 4.2.3 - Comparación visual de resultados, con y sin búsqueda local

5. Conclusiones

En [13] se resuelve el problema aplicando Algoritmos Genéticos con función objetivo más simple. Los resultados informados muestran un desvío del 65% respecto del óptimo con un tiempo computacional de 330 minutos.

En este trabajo, de acuerdo a los tests realizados, es posible obtener imágenes de buena calidad en tiempos de ejecución cercanos al 37% del tiempo usado por el algoritmo determinístico, con un desvío del óptimo del 25.6%. El tiempo de ejecución y la calidad de compresión obtenida, nos permiten sostener que la metaheurística ACO puede ser aplicada en forma exitosa al problema de la compresión fractal de imágenes. Además, nuestro algoritmo es otra alternativa a usar en la compresión de imágenes.

Para lograr "buenos resultados", hubo que considerar los siguientes factores:

- *Número de hormigas*: de acuerdo a [4], recomienda usar 10 (ó *n* hormigas, siendo n, en este caso, el número de bloques de dominio). En nuestro caso, se usó el primero. Debe tenerse en cuenta que cada hormiga almacena su camino y esto afecta al uso de memoria, por lo que es recomendable no usar demasiadas hormigas.
- *Iteraciones*: depende del problema. Está íntimamente relacionado con el número de hormigas, y también por el número de hormigas que depositan feromona. Debe permitir lograr un resultado "aceptable" en corto tiempo; además debe permitir mejorar el óptimo inicial propuesto. Es decir, pocas iteraciones, malos resultados; muchas iteraciones, buenos resultados.
- *Factor de evaporación* (ρ): según [6], usa 0.5 para resolver el problema del viajante. En este trabajo, se obtuvieron buenos resultados con valores cercanos a 0,7. Lo importante a tener en cuenta con este parámetro es el hecho de que valores cercanos a 1, obliga a la exploración de nuevos espacios de soluciones mientras que valores cercanos a 0, explora el entorno de la solución actual.
- *Solución inicial*: si bien, tanto en [4], [6] y [2] se hace mención a la feromona inicial aplicada a cada par (i,j) como una función que relaciona el número de hormigas con una solución inicial, no hay precisiones sobre la distancia de ésta con la deseada. Es vital tener en cuenta esta situación. Para este trabajo, se ha determinado que la solución inicial debe estar entre 1.7 y 2.5 veces del valor óptimo. Valores superiores provocan, un lento descenso hacia una buena solución; en tanto, valores inferiores pueden desembocar en la infactibilidad del problema. Colonia de hormigas no requiere soluciones iniciales de alta calidad, pero sí requiere no estar muy lejos.
- *Búsqueda Local*: en [4], menciona la importancia de la información heurística si no se aplica búsqueda local a un problema de TSP. En este trabajo, se considera un factor predominante que es el uso de algoritmos de búsqueda local para evitar estancamiento (stagnation) y converger rápidamente a soluciones razonables. Cuando no se usó búsqueda local, las soluciones obtenidas estaban muy alejadas de las esperadas (el doble del mejor óptimo con búsqueda local). Si bien puede achicarse esta diferencia, implica un mayor número de iteraciones provocando mayor tiempo de ejecución y no asegurando soluciones razonables.
- *Información heurística*: en [3] (como en la mayoría de los trabajos de Colonia de Hormigas) considera como información heurística la recíproca de la distancia di_j, es decir la distancia entre el par (i,j). Sin embargo,

trabajos como [10] considera como información heurística, al tamaño del ítem j. En nuestro caso, se consideraron tres alternativas, siendo la tercera la que produjo mejores resultados:

- Diferencias de medias: este alternativa no produjo buenos resultados, ya que no es cierto que menores distancias entre los bloques de rangos y de dominios, conducirán a menores valores de la función objetivo.
- Frecuencias(i,j): consistía en registrar el número de veces que el bloque de dominio j era usado para comparar con el bloque de rango i. La falla de esta opción, radicó en el hecho de que era posible que las hormigas eligieran malos bloques (mayor error) y por esto sigan siendo elegidos en contraste con buenas selecciones pero con baja frecuencia.
- Error de selección(i,j): como se indicó anteriormente, es el error provocado por elegir el bloque de dominio j para comparar con el bloque de rango i. De esta forma, las frecuencias no son tenidas en cuenta. Además, la información es cambiante, no fija como en la mayoría de los casos resueltos con esta metaheurística. Básicamente, el error de selección es equivalente al costo de selección.

Por último, existen varios caminos de mejoras que puede aplicarse a fin de obtener resultados de buena calidad en menor tiempo:

- *Categorización de bloques*: tal como lo realiza [8] para resolver el problema en forma determinística. Sin embargo, debe considerarse este tiempo de procesamiento previo a la resolución "real" del problema.
- *Búsqueda local*: mejorar los caminos obtenidos por las hormigas a través de diferentes algoritmos de búsqueda que permitan converger a buenas soluciones.
- *Trabajo cooperativo*: formar grupos de hormigas que busquen clusters en la imagen que puedan usarse para la compresión, aumentando de esta forma la tasa de compresión.

Agradecimientos

A Irene Loiseau y Silvia Ryan por sus comentarios y correciones, que ayudaron a mejorar la calidad del trabajo.

Referencias

[1] Barnsley, M., *Fractal Image Compression*, A.K. Peters, 1993, págs. 89-116.
[2] Bonabeau, E., Dorigo, M. & Theraulaz,G., *Swarm Intelligence*, Oxford, 1999, págs. 32-77.
[3] Bullnheimer B., Hartl, R. & Strauss, C., *Applying the Ant System to the Vehicle Routing Problem*, INRIA, 1997, págs. 1-10. Disponible en www.aco-metaheuristic.org.
[4] Dorigo, M. & Stutzle, T., *Ant Colony Optimization*, MIT, 2004, págs. 12-117.
[5] Dorigo, M. & Gambardella, L., *Ant Colonies for the Travelling Salesman Problem*, BioSystems, 1997, págs. 1-7. Disponible en ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.15-BIOSYS97.ps.gz
[6] Dorigo, M., Di Caro, G. & Gambardella, L., *Ant Algorithms for Discrete Optimization*, Tech Report IRIDIA, 1999, págs. 7-20. Disponible en ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.23-alife99.ps
[7] Fisher, Y., *Fractal Image Compression*, Siggraph, 1992, págs. 2-14.
[8] Jacquin, F., *Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations*, IEEE Transactions on Signal Processing, 1992, págs. 2-7.
[9] Hamzaoui, R., Saupe, D. & Hiller, M., *Fast Code Enhacement with Local Search for Fractal Compression*, Leipzig University, 2000, págs. 1-3. Disponible en http://www.inf.uni-konstanz.de/~hamzaoui/index.shtml.en
[10] Levine, J. & Ducatelle, F., *Ant Colony optimization and local search for the bin packing and cutting stock problems*, Tech Report, 1997, págs. 1-10. Disponible en http://www.idsia.ch/~frederick/levine-jors03.pdf
[11] Lu, N., *Fractal Image Compression*, Academic Press, 1997, págs. 2-82.
[12] Ryan, S., Martinez, C. & Ryan, G., *Problema de Asignación de Aulas en la Universidad Nacional de Salta*, Sobrapo, 2004, págs. 3-5.
[13] Vences, L. & Rudomin, I., *Genetic Algorithms for the Fractal Image and Image Sequence Compression*, Visual 97, 1997, págs 1-6. Disponible en http://rudomin.cem.itesm.mx/~rudomin/

HERRAMIENTA VISUAL DE PROCESAMIENTO DE IMÁGENES

Claudia A. Barbosa
Fundación Universitaria San Martín, Programa Ingeniería de Sistemas,
Bogotá, Colombia
clauanbar@hotmail.com

y

Heidi P. Morales
Fundación Universitaria San Martín, Programa Ingeniería de Sistemas,
Bogotá, Colombia
heidip48@yahoo.com

Abstract

This paper resumes the development of a software to design image processing algorithms graphically. In this software, each operation is represented by a box and the relationship with other operations is made by connecting graphically the outputs of the boxes to the inputs of the other ones. Some boxes were implemented to model control structures (conditionals and loops) and procedures. The software has two execution modes: step by step and automatic. The software uses the library JAI (Java Advanced Imaging) as its processing engine.

Keywords: Digital image processing, graphical editor, visual programming, JAI.

Resumen

Este trabajo resume el desarrollo del proyecto de creación de una herramienta visual para el diseño de algoritmos de procesamiento de imágenes. En la herramienta, cada operación se representa con una caja y la relación con otras operaciones se realiza mediante la conexión gráfica de las salidas de unas cajas con las entradas de otras. Se implementaron cajas para modelar estructuras de control (condicionales y ciclos) y procedimientos. El software tiene dos modos de ejecución: paso a paso o automático. La herramienta usa como motor de procesamiento la librería JAI (Java Advanced Imaging).

Palabras clave: Procesamiento digital de imágenes, Editor gráfico. Programación visual, JAI.

I. INTRODUCCIÓN

El procesamiento de imágenes se puede usar en muchas aplicaciones útiles para resolver problemas que se presentan en diversas áreas como la medicina, biología, sistemas de información geográfica y en la vida cotidiana.

Los programadores de este tipo de soluciones informáticas tienen varias alternativas disponibles para desarrollar software: programar sus propios algoritmos, usar librerías de procesamiento de imágenes, usar editores gráficos de procesamiento de imágenes o una combinación de las anteriores.

El uso de librerías es una de las estrategias más usadas, porque permite la reutilización de código que ya ha sido probado y optimizado en los nuevos programas de procesamiento, sin embargo, tiene el inconveniente de que el tiempo de desarrollo, optimización y realización de pruebas es usualmente lento y que no todas las librerías son fáciles de adquirir por sus costos.

La anterior estrategia se suele combinar con el desarrollo de algoritmos especializados muy puntuales que no pueden ser cubiertos con las funciones implementadas en la librería. Lo ideal es que los algoritmos que tenga que desarrollar el programador sean de complejidad baja para hacer que el proceso de optimización de los algoritmos sea más rápida de realizar.

La última estrategia evaluada fue la programación gráfica de procedimientos de procesamiento de imágenes. Esta estrategia tiene ventajas y desventajas. La ventaja más importante es la facilidad que ofrece para hacer pruebas y para realizar programas en un tiempo menor al utilizado en las otras estrategias. Las desventajas radican en que se dificulta la inclusión de algoritmos personalizados en los programas y todos los problemas que se presentan comúnmente en la programación visual, como la dificultad de hacer mantenimiento y documentación del código a medida que crece el tamaño de los programas.

En la revisión bibliográfica se encontró una gran variedad de librerías gráficas tanto en software libre como en software comercial. Sin embargo, en software de libre distribución pocas herramientas ofrecen editores de programación gráficos. Por esta razón, se tomó la decisión de hacer un editor que usara como base una librería de libre distribución para hacer programas de procesamiento de imágenes.

II. DESCRIPCIÓN

El software se compone de un editor gráfico para el desarrollo de algoritmos de procesamiento de imágenes de forma gráfica. Los requerimientos funcionales más importantes incluyen:

- La facilidad de uso basado en la experiencia que tienen los diseñadores de algoritmos de procesamiento
- La extensibilidad, que en este caso se traduce en qué tan fácil es incluirle nuevas operaciones al menú
- La facilidad de instalación
- La minimización de las dependencias de librerías
- La eficiencia en la ejecución de los algoritmos, por lo cual se requiere que el software use como motor una librería probada de procesamiento de imágenes.
- El software debe ser de libre distribución
- Debe tener dos modos de ejecución: paso a paso y automático para correr todo el algoritmo representado gráficamente.
- Debe mostrar los errores e inconsistencias en el proceso de creación de los programas para ayudar a los programadores a depurar sus procesamientos.

Para cumplir con lo anterior, el primer paso que se tomó fue la revisión de las herramientas disponibles en el mercado para determinar cuáles se podían usar en el desarrollo.

III. HERRAMIENTAS EXISTENTES PARA PROCESAMIENTO DE IMAGENES

3.1. Aplicativos de procesamiento de imágenes

En el mercado hay una gran variedad de aplicativos. A continuación se describen brevemente tres de ellos.

- **Khoros – VisiQuest:** Es un ambiente de integración y construcción de programas de procesamiento de imágenes que incluye una amplia gama de herramientas de desarrollo junto con una colección extensa de algoritmos de procesamiento y manipulación de datos. Ofrece un ambiente de programación visual llamado Cantata. Este ambiente proporciona la interfaz que permite unir las salidas de los algoritmos de procesamiento a los siguientes de forma gráfica. En Cantata, los programas visuales se crean como grafos dirigidos, en donde cada algoritmo de procesamiento es un nodo del grafo y cada arco dirigido representa el paso de parámetros de las salidas de unos algoritmos a las entradas de otros. Es una herramienta muy completa pero tiene un costo elevado [1].
- **MATLAB:** Es una herramienta basada en procesamiento matricial que permite expresar en forma compacta operaciones matemáticas; esto hace fácil e intuitivo efectuar procesamiento de imágenes y operaciones de análisis tales como la Transformada de Fourier, filtrado 2-D, morfología binaria y manipulación geométrica mediante su toolbox de procesamiento de imágenes. Tiene una herramienta de programación gráfica, Simulink, con la cual se pueden diseñar algoritmos de procesamiento de imágenes usando diagramas de bloques, de forma similar a como lo hace la herramienta Cantata de Khoros – VisiQuest. Es una herramienta muy completa pero tiene un costo elevado [6].
- **AreaSat:** Está dirigido al procesamiento de imágenes satelitales y tiene funciones para el manejo de información geográfica. Tiene un alcance limitado a aplicaciones de información geográfica [3].

3.2. Librerías de procesamiento de Imágenes

En el mercado se encontraron entre muchas otras las siguientes librerías:

- **VTK:** Es una librería poderosa de código abierto para procesamiento de imágenes. Aunque está diseñada para ser utilizada en programas hechos en C++, tiene extensiones para trabajar con otros lenguajes como Java, Python, Visual Basic, etc. El sitio Web está en http://www.vtk.org
- **ImgSource:** es una librería no libre que se puede usar en Windows en cualquier lenguaje de programación. Tiene cerca de 350 operaciones de procesamiento. El sitio Web está en http://www.smalleranimals.com/isource.htm
- **VIPS – nip2**: Es una librería de software libre que se puede extender con algoritmos hechos en C. Tiene un editor gráfico llamado nip2. El sitio Web está en http://www.vips.ecs.soton.ac.uk/
- **IPL98 – Image Processing Library 98**: Está hecho en C/C++ y se puede integrar con programas hechos en C/C++ por los usuarios. El sitio Web está en http://www.mip.sdu.dk/ipl98/.
- **JIU - *Java Imaging Utilities***: Es una librería de software libre para ser empleada en programas en Java diseñados por el usuario. El sitio Web está en http://jiu.sourceforge.net
- **JAI – Java Advanced Imaging**[9]: es una librería de Sun Microsystems que contiene una gran variedad de algoritmos de procesamiento. El usuario debe hacer programas en Java que la utilicen como motor de procesamiento. Tiene 46 operaciones de procesamiento. El sitio Web está en http://java.sun.com/products/java-media/jai/.

IV. ARQUITECTURA DEL SISTEMA

Después de estudiar el software existente en el mercado se tomó la decisión de usar como motor de procesamiento la librería JAI (Java Advanced Imaging) dado que es la desarrollada por Sun Microsystems Inc y es de fácil consecución, instalación y distribución; sin embargo, la arquitectura del software desarrollado tiene en cuenta que debe ser fácilmente acoplable a cualquier otra librería.

El IDE seleccionado fue Eclipse y se tomó la decisión de no utilizar librerías diferentes a las provistas por Java 1.5. La excepción a esta decisión estuvo en la generación de histogramas, puesto que se necesitaba una librería que generara gráficos en línea, por lo cual se utilizó JFreeChart [4]. Esto se hizo para hacer el software lo más fácil distribuible e instalable posible.

Esta arquitectura consta de dos paquetes: Interfaz y Datos.

- El paquete "Interfaz" contiene la interfaz gráfica. Aquí se diseñan los componentes, el menú, el estilo de cada caja y sus propiedades y los enlaces. Este paquete tiene una clase principal que incluye todas las opciones del menú y a su vez contiene el área de trabajo donde se visualizan los componentes (cajas, enlaces y características). Las cajas contienen enlaces de entrada y de salida y a su vez tiene asociadas características o parámetros específicos de cada operación de procesamiento.

- El paquete "Datos" modela el funcionamiento del sistema, la conexión a los algoritmos de JAI utilizados en cada caja, el funcionamiento de ciclos y procedimientos, y como se enlazan las cajas entre sí y el modo de ejecución. Cada algoritmo JAI, corresponde a una clase que hereda de la clase "Caja". Dependiendo del tipo, se le asigna el número de enlaces de entrada (parámetros de entrada del filtro) y enlaces de salida (Resultados después de la ejecución). Los procedimientos y ciclos también heredan de la clase caja pero estos no contienen un algoritmo en JAI sino una secuencia de cajas enlazadas.

El siguiente diagrama muestra las relaciones entre las diferentes clases del sistema:

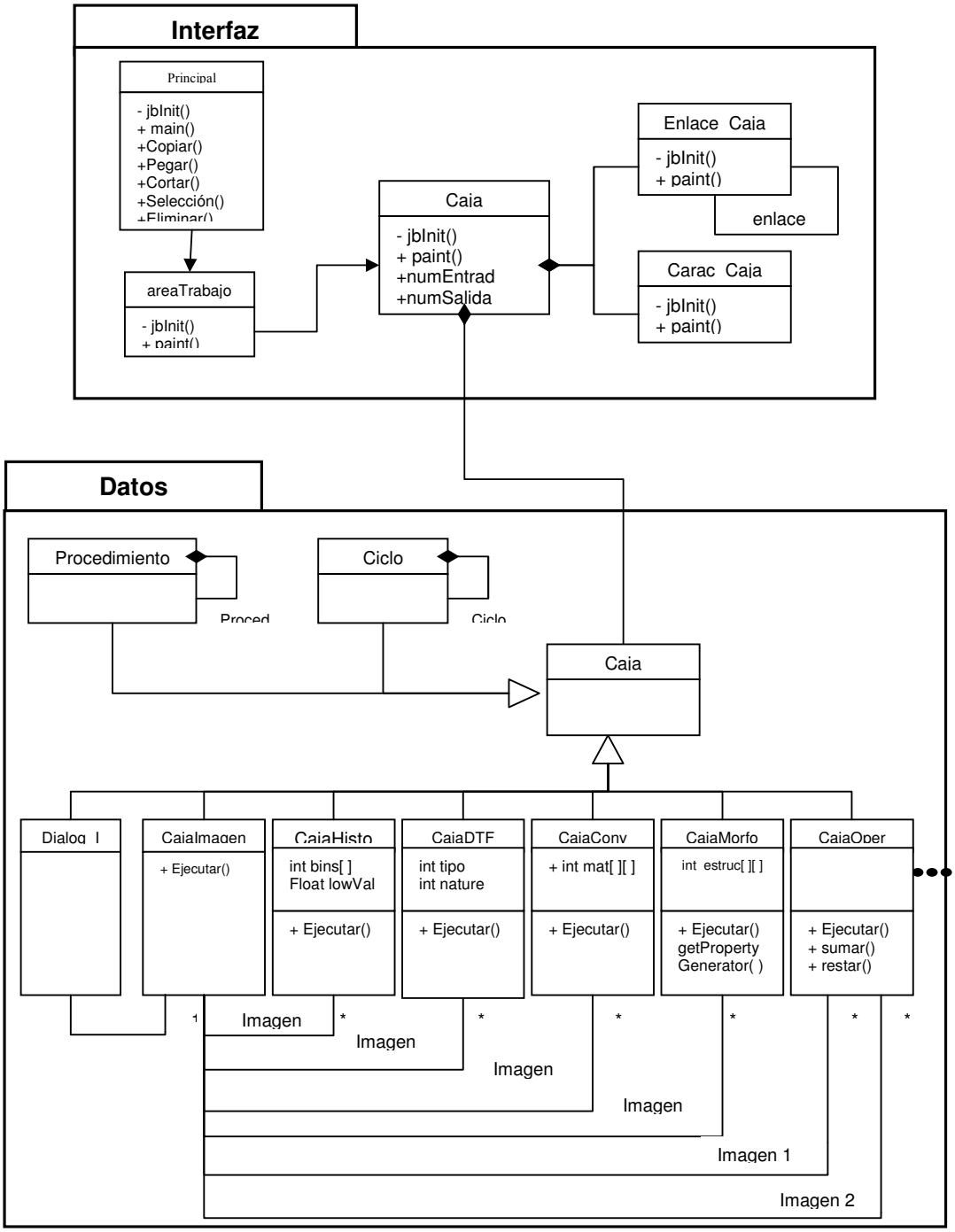


Figura 1. Arquitectura del Software para procesamiento de Imágenes [2]

La arquitectura fue diseñada para que el software fuera fácilmente extensible mediante la creación de nuevas operaciones por parte del usuario. Los pasos que se deben seguir si se quisiera crear una nueva caja con una nueva operación son los siguientes:

1. Se debe crear una clase en \khoros\src\khoros\modelo que herede de la clase Caja
2. Agregar los enlaces de entrada y de salida estos enlaces son de tipo Enlace_Caja:

private Enlace_Caja salida = new Enlace_Caja(posx, posy, tipo);

El tipo es para especificar si el enlace es de entrada o de salida.

3. Agregar características si el filtro lo necesita, estos son de tipo Caract_Caja:

   private Caract_Caja prop = new Caract_Caja(posx, posy, tipo);

   Como parámetros a esta clase le entran las posiciones y que tipo es "P" significa que se va crear un panel, esto es para el caso del ciclo y el procedimiento. "F" significa que es para crear un *frame* con parámetros de entrada para filtro.
4. La caja siempre tendrá una componente de tipo *JToggleButton* que es botón de ejecutar el filtro:

   private JToggleButton run = new JToggleButton();

5. La clase que está creando debe tener obligatoriamente tres métodos:
   - *public void eventos()*: Llama al *frame* que se necesita para tener los parámetros de entrada del filtro.
   - *public void eventos1()*: Llama el panel, pero este evento sólo se utiliza en ciclos y procedimientos, de lo contrario va en blanco.
   - *public void ejecutar()*: Es el método donde se encuentra el desarrollo del filtro. Este filtro debe retornar una imagen de tipo *RenderedOp*.
6. Finalmente se agrega al menú para ser operado. Esto se realiza en la clase de KhorosFrame, se adiciona al menú de filtros. El evento de este filtro va a crear una caja nueva de la clase que se creo con el filtro deseado.

Para cambiar de JAI a otra librería de procesamiento, se deben cambiar los métodos "ejecutar" de las clases que representan cada operación para que hagan el llamado a la librería deseada.

V. DISEÑO DE LA INTERFAZ

Se tomó la decisión de hacer un diseño de interfaz gráfica muy similar al ofrecido por *Cantata*, dado que es fácil de comprender por los usuarios y facilita el aprendizaje de los programadores familiarizados con el ambiente de programación ofrecido por *Khoros*[1] (ahora llamado *VisiQuest*).

Cada operación, proceso o filtro está representado en la interfaz por una caja. En la figura 1 se muestran los componentes de una caja y su ubicación en la interfaz:

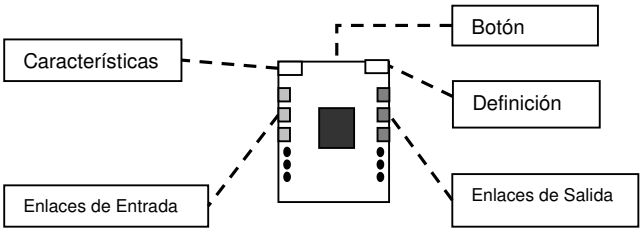


Figura 2.Componentes de la caja [2]

La interfaz permite definir para cada tipo de caja los parámetros o características necesarias para su ejecución. Al dar clic sobre el botón de características aparece la caja de diálogo que permite personalizar los parámetros de la operación o filtro.

En la zona de enlaces de entrada y salida, la interfaz permite conectar a la caja con otras. La mayoría de las cajas utilizan como entradas las imágenes generadas como producto de etapas previas del procesamiento. Cada caja tiene un número diferente de entradas y salidas pues esto depende de la operación que se esté realizando.

Los enlaces en las cajas se crean dependiendo de los parámetros de entrada y salida que contenga la operación. Los enlaces *Enlace_Caja()* en el paquete de interfaz, contienen un atributo del mismo tipo. Cuando se está creando un enlace entre dos cajas, el enlace de tipo entrada guarda en ese atributo la especificación del enlace con el está conectado.

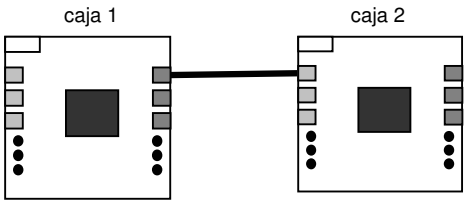
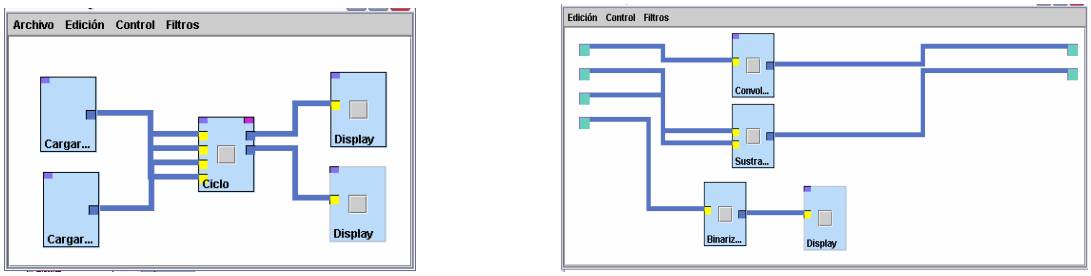


Figura 3: Enlace entre cajas [2]

Al tener un enlace entre la caja 1 y la caja 2, el enlace de entrada de caja 2 almacena todas las características del enlace de salida de la caja 1. Este diseño esta pensado para facilitar el modo automático de ejecución.

Las cajas de procedimientos y ciclos están compuestas por otras cajas. La definición de un procedimiento o ciclo se puede obtener al dar la opción de "definición" de la caja. Aparecerá el algoritmo gráfico asociado a la caja de procedimiento o ciclo. En estos casos, el número de entradas y salidas dependerá del algoritmo implementado. Adicionalmente, en cada ciclo se debe ingresar el número de veces que se desea ejecutar y asignar cada una de las salidas del ciclo a las entradas para evitar que dé resultados equivocados. Al ejecutarse en cada iteración se reemplazarán los parámetros de entrada al ciclo por las salidas que estén conectadas a ellos.

En el siguiente gráfico se muestra un ejemplo que contiene un ciclo y la definición del algoritmo que se ejecuta varias veces:



a) b)

Figura 4. a) Ejemplo ciclo b) Algoritmo dentro del ciclo

Los ciclos y procedimientos contienen *N* número de enlaces de entrada y de salida, este número de enlaces varía dependiendo de la secuencia de filtros que se esté realizando en el momento. Los ciclos y procedimientos tienen unos puntos verdes llamados referencias que lo que hacen es conectar las entradas de la caja ciclo a las entradas de las cajas internas.

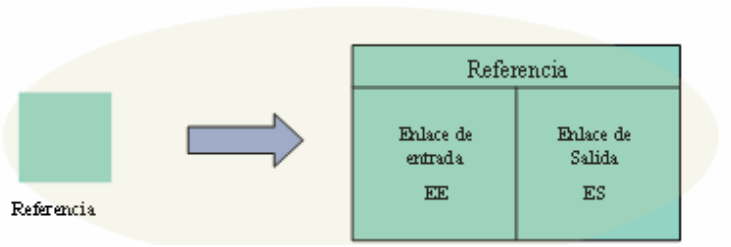


Figura 5. Diseño de Referencia para Ejecución del Ciclo

En el caso de los ciclos, adicionalmente es posible conectar las referencias de salida a las de entrada. Todo esto se hace en las características de la referencia con el fin de que el programa se mantenga como un grafo dirigido sin ciclos.

Cada caja tiene un botón de ejecución que está en la mitad de cada una. Al ejecutar una operación, se realiza el procesamiento con la ayuda de la librería JAI (Java Advanced Imaging) [9].

Los parámetros que se configuran en las diferentes operaciones son flotantes, enteros o archivos de texto en los cuales se digitan los datos de las matrices estructurantes separados por comas. Por ejemplo una matriz de 3X3 sería así:

1.1	1.1	1.1
1	-8.6	1
1.1	1.1	1.1

Figura 6. Matriz estructurante de 3 x 3

Se guarda en archivos con el siguiente formato:

Estructurante.txt

```
1.1, 1.1, 1.1
1,   8.6, 1
1.1, 1.1, 1.1
```

Figura 7. Archivo de la matriz estructurante

Algo similar se tiene en los estructurantes de las operaciones de morfología matemática con la diferencia de que los valores son enteros.

Casi todas las cajas manejan el mismo tipo de dato de entrada y salida: imagen, sin embargo, en algunas operaciones la salida no es de este tipo, por lo cual se hace necesario un validado de tipos que se activa al conectar las entradas de una caja con las salidas de otra. En caso de no coincidir el tipo (e.g. se esperaba una imagen y se recibe un flotante) se obtiene un mensaje de error y no se permite el cambio.

VI. EJECUCION DE LOS PROGRAMAS

Al ejecutar una caja es probable que las entradas no estén listas para ser utilizadas porque pueden ser salidas de cajas que no se han ejecutado aún. Por esta razón, fue necesario tomar decisiones en la forma en que el sistema debería resolver este problema.

Existen dos modos de ejecución del programa: paso a paso y automática. Cuando se ejecuta paso a paso sólo se ejecuta una caja siempre y cuando estén listas las entradas (i.e. ya se habían calculado con anterioridad las entradas).

El modo de ejecución automática de un algoritmo realizado en el sistema, está diseñado de la siguiente manera:

La operación se realiza de forma recursiva, se revisa qué caja no tiene listas las salidas referenciadas en las entradas de la caja que se está ejecutando. Esto se hace hasta llegar a cajas que tengan todas sus entradas disponibles, ya sea porque no tienen entradas o porque ya se han ejecutado previamente las cajas predecesoras. A partir de estas cajas se comienzan a ejecutar todas las cajas necesarias para tener disponibles las entradas en la caja que se quería ejecutar originalmente, con lo cual ya es posible la ejecución de la operación.

Esto hace que cuando el usuario quiera ejecutar una operación, se ejecuten varias puesto que el software evalúa las dependencias y ejecuta lo que sea necesario para realizar la acción solicitada.

A continuación se muestra un ejemplo de la forma en que se ejecutan las operaciones.

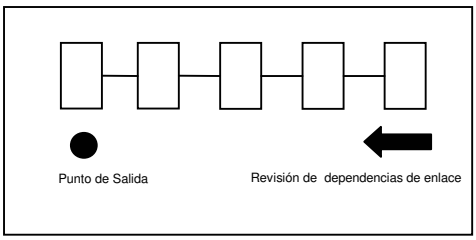


Figura 8. Modo de Ejecución

En este caso, el sistema asume que el comienzo del proceso es la caja que en no hay entradas (la primera de la izquierda. A partir de este punto empieza a recorrer la ruta de enlaces ejecutando cada caja y pasando como parámetros las salidas a las entradas de la siguiente caja, hasta llegar a la caja con la que se inició el proceso.

Se pueden presentar también el siguiente caso en la ejecución:

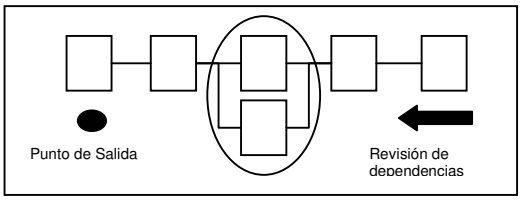


Figura 9. Modo de Ejecución

En este caso de la figura anterior se puede tener el conflicto de cual caja debe ser ejecutada primero: la de arriba o la de abajo. El programa supone que las dos cajas tienen la misma prioridad y las ejecutará en orden de hasta llegar al final del algoritmo.

Es claro que este esquema de ejecución no funcionará cuando existan dependencias cíclicas. Si se observa el modelo de programación, si el desarrollador hace un programa con dependencia cíclica, no habrá forma de ejecutarlo ni manual ni automáticamente, porque nunca se tendrán las entradas disponibles en ninguna de las cajas involucradas para poder ejecutarlas. Por esta razón, el algoritmo de ejecución se da cuenta de la existencia de dependencias cíclicas, aborta la operación y muestra un mensaje de error para que el desarrollador arregle el problema.

Hay que diferenciar una dependencia cíclica de un ciclo. La forma como se implementaron los ciclos permite que se eviten las dependencias cíclicas puesto que en ningún caso se permite que el grafo dirigido tenga ciclos y el número de iteraciones se determina mediante las características de la caja de ciclos.

En el momento en que se realiza la primera ejecución de un ciclo, las imágenes que procesa son las que le llegan de las cajas de afuera y ésta es almacenada en el enlace de entrada de la referencia de entrada de la caja que representa el ciclo (ver apartado V). Cuando se realiza la siguiente iteración, la imagen procesada queda almacenada en el enlace de entrada de la referencia de salida y éste la pasa al enlace de salida de la referencia de entrada, asegurando de esta forma que la imagen que se esta ejecutando por segunda vez es la imagen ya modificada por la secuencia de operaciones definidas dentro del ciclo.

Uno de los problemas que se presentan en la ejecución del ciclo es cuando el número de enlaces de entrada es diferente al número de enlaces de salida. En la parte de características del ciclo se le asigna a las referencias de entrada la referencia de salida que se desea para la siguiente iteración. Esta asignación la determina el usuario al igual que el número de iteraciones que se desea ejecutar el ciclo.

Cuando se ejecuta una sola caja dentro del ciclo, la ejecución llega hasta ese punto y solo lo hace una vez.

La diferencia entre la ejecución de los procedimientos y los ciclos, es que los procedimientos sólo se ejecutan una vez y la primera imagen procesada es la que llega a las cajas fuera del filtro. Aquí no necesitamos la asignación de referencias ni el número de iteraciones.

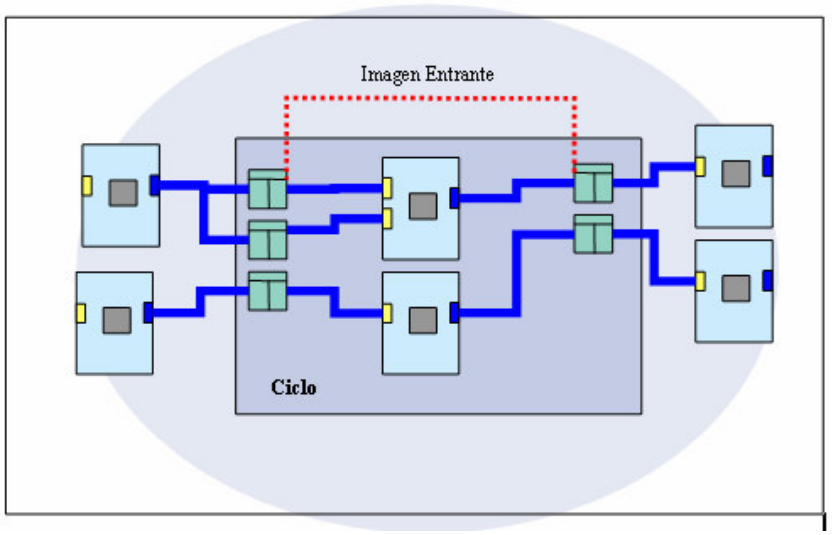
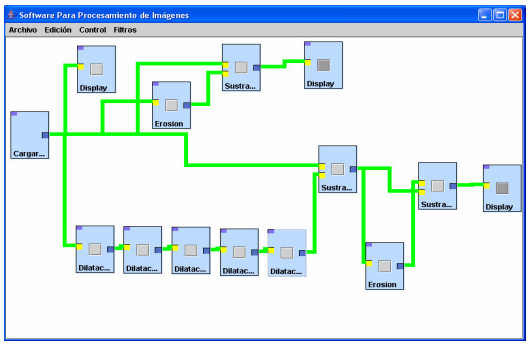


Figura 10. Diseño del Ciclo en el Software para Procesamiento de Imágenes

En caso que el usuario intente ejecutar una caja que está dentro de un ciclo o procedimiento, el algoritmo de ejecución realiza las operaciones de procesamiento hasta la caja en cuestión y todas las cajas predecesoras.

A continuación se muestra una imagen de un algoritmo realizado en el Software en el que se utiliza las operaciones de dilatación y de erosión, que se utilizan para engrosar o disminuir los bordes de las figuras respectivamente, según el estructurante que se aplique se redondean los bordes y al aplicar varias veces la dilatación el resultado de la imagen es totalmente oscura como se muestra en la siguiente figura.

a)



b)

c)

d)

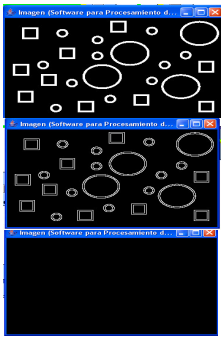


Figura 11. a) Ejemplo en el Software para Procesamiento de Imágenes b) Imagen original c) Resultado después de aplicar el filtro Morfológico de erosión. d) Resultado después de aplicar varias veces el Filtro Morfológico de dilatación

VII. OPERACIONES

Después de haber evaluado las operaciones y filtros disponibles en la herramienta de procesamiento de imágenes *Khoros-VisiQuest* y MATLAB y de haber estudiado las operaciones y filtros disponibles en la librería JAI, se implementó la interfaz con las operaciones que se describen a continuación:

Convolución/Correlación: Está operación permite implementar las operaciones de convolución y correlación sobre una imagen [5].
Los parámetros de esta operación son: la matriz de datos que representa la imagen y el estructurante que es una matriz de tamaño N*N de flotantes. En la interfaz, la imagen corresponde a la primera entrada de la caja de convolución/correlación y el estructurante se determina en la caja de diálogo de características mediante el uso de un archivo de texto que tiene el siguiente formato.

Matriz.txt

```
1, 1, 1
1, -8, 1
1, 1, 1
```

Figura 12. Estructurante para Detección de Bordes

Histograma: Es una operación que permite al usuario generar el histograma a una imagen determinada, es decir, una gráfica en la que se muestra la frecuencia de aparición de cada uno de los colores en la imagen [5].

Se utilizan los siguientes parámetros:

- numBins es un arreglo de enteros, que especifica el número de colores utilizados por cada banda de la imagen se declara como {256, 256, 256} en el caso que la imagen tenga 3 bandas (RGB) . El número de elementos en el arreglo es igual al número de bandas de la imagen.
- lowValue Un arreglo de flotantes, de cada elemento se especifica el nivel de gris o color más bajo que podría ser comprobado por una banda de la imagen, se declara de acuerdo al numBins anterior como 0.
- highValue Un arreglo de flotantes, cada elemento se especifica el nivel de gris o color mas alto que podría ser comprobado por una banda de la imagen, se declara de acuerdo al numBins anterior como 256.

Recibe como entrada la imagen y en las características se puede modificar los valores de numBins, lowValue y highValue. Al ejecutar aparece el reporte del histograma de la imagen.

Para la generación de los histogramas se utilizó la librería de generación de reportes gráficos JFreeChart [4] que permite hacer que la matriz de datos que se genera al aplicar la operación del histograma, se pueda convertir en una imagen, para que se pueda mostrar utilizando la librería JAI [9], ya que esta operación no genera imagen de salida.

Operaciones relacionadas con la Transformada de Fourier: Se puede aplica la Transformada Rápida Directa e Inversa de Fourier en 2 dimensiones a una imagen [5]. Esto es útil para aplicar operaciones en el dominio de la frecuencia para luego retornar al dominio espacial.

Mediante la manipulación de las características de la caja de Transformada Directa de Fourier, se puede obtener como salida la parte real o la imaginaria en forma de imagen cuando se está trabajando en el dominio de la frecuencia.

Morfología Matemática: Se implementaron las dos operaciones básicas de la morfología matemática disponibles en JAI:

- **Dilatación:** esta operación permite efectuar la operación de dilatación de morfología matemática [5]. JAI detecta si la imagen está en colores, en tonos de grises o en blanco y negro y dependiendo de ello realiza aplica la operación.

  Recibe como entrada la imagen a procesar y se configura el archivo texto donde se describe el estructurante. El archivo tiene el formato .txt utilizado para la convolución. La salida es la imagen procesada

- **Erosión:** esta operación permite efectuar la operación de erosión de morfología matemática [5]. JAI detecta si la imagen está en colores, en tonos de grises o en blanco y negro y dependiendo de ello realiza aplica la operación.

  Al igual que en la dilatación, recibe como entrada la imagen a procesar y se configura el archivo texto donde se describe el estructurante que tiene el mismo formato de los estructurantes usados en la dilatación.

Con estas operaciones se pueden formar otras más como la apertura, cerramiento, Hit or miss, etc [5].

Operaciones aritméticas: Se tienen disponibles las cuatro operaciones básicas entre dos imágenes, estás se tratan como operaciones entre matrices de píxel a píxel sumando, restando, multiplicando o dividiendo según sea el caso [5]. En

caso que el color quede por fuera del rango válido, JAI hace el ajuste correspondiente. En el caso de la división JAI tiene en cuenta las posibles divisiones por cero.

Operaciones lógicas: Se tienen cuatro operadores lógicos AND, OR, XOR, NOT, se realiza las operaciones entre las matrices de las dos imágenes píxel a píxel, sin importar si son de diferente en tamaño o tipo de datos. [5]

Operaciones geométricas: Se utilizan tres operaciones que son la traslación, rotación y escalar, estas operaciones permiten al usuario digitar lo que desee trasladar una imagen sobre el eje *x* o eje *y* y el ángulo que la desee rotar, estas operaciones se realizan entre la matriz de la imagen y cada uno de los números digitados por el usuario. [5].

Transponer: Permite tomar una imagen e invertirla de forma horizontal, vertical, diagonalmente o rotar 90°, 180° y 270° sobre el eje x o el eje y. [5].

Operadores Relacionales: Se encuentran dos operadores el máximo y el mínimo permitiendo resaltar en una imagen el color más alto o más bajo, de acuerdo a las bandas que maneje cada imagen. [5].

Combinado de banda: Esta operación recibe una imagen a color y la pasa a tonos grises [9].

Copiado de banda: Existen tres tipos que son Banda2, Banda3 y Banda4, que cada uno resalta en una imagen un nivel de color especifico, la primera grises, y las demás de acuerdo al mayor nivel de color que tenga la imagen [9].

Binarización: Esta operación permite convertir la matriz de una imagen a dos colores: blanco y negro, pero para convertir una imagen de color es necesario primero convertirla a gris, esto lo realiza usando un valor de umbral de binarización [5].

Gradiente: En este filtro utiliza un estructurante horizontal y otro vertical, dados en archivos de texto .txt, entonces cada uno permite resaltar los objetos lineales o bordes orientados en una cierta dirección (p.e. horizontal, vertical o diagonal, esta operación consiste en calcular la diferencia que existe entre la primera derivada de un píxel y la de sus vecinos [5].

Mediana: Se utilizan tres tipos de Mascaras *Cuadrada*, *Cruz* y *X*, que permiten eliminar ruido en las imágenes, suavizar los bordes y las áreas de cada color se vuelven más evidentes [9].

VIII. RESULTADOS

Antes de hacer comparaciones entre esta herramienta y otras disponibles en el mercado, hay que resaltar que la evaluación sólo tiene en cuenta las operaciones comunes de procesamiento para los programas, puesto que, como ya se indicó, se tiene la restricción de que JAI tiene un conjunto de operaciones menor a las herramientas comerciales. Se evaluaron los siguientes aspectos:

Número de operaciones disponibles: JAI tiene 46 operaciones disponibles. Khoros – VisiQuest tiene más 500 y el toolbox de MATLAB tiene cerca de 300. Aunque la herramienta tiene menos operaciones, está diseñada para cambiar de librería fácilmente, con lo cual se incrementaría el número de operaciones rápidamente.

Costos: la herramienta desarrollada es software libre. Las herramientas comerciales que tienen editor de programas (Cantata y Simulink entre otros) tienen un costo elevado en el licenciamiento.

Tiempo de ejecución y uso de memoria: la ejecución y uso de memoria de los algoritmos depende casi en su totalidad de la implementación de las operaciones que tenga la librería. En este caso, el tiempo de ejecución y uso de memoria de los algoritmos depende de JAI y por lo tanto, si se va a comparar con otras herramientas, se debe comparar dicha librería con la herramienta. Al establecer un paralelo entre JAI, Khoros - VisiQuest y MATLAB se encontró que tienen tiempos de ejecución y uso de memoria similares. Se realizaron 20 algoritmos utilizando intensivamente las operaciones de convolución, operaciones morfológicas, FFT e histogramas con las tres herramientas donde su tiempo de ejecución y uso de memoria tenían una diferencia mínima, además los resultados del procesamiento fueron iguales en cada una de ellas.

Facilidad de uso: Tanto Khoros- VisiQuest, MATLAB y la herramienta desarrollada tienen una facilidad de uso similar. Esto se debe a que las dos primeras eran la que tenían mayor facilidad de uso y por esta razón, se quiso implementar la interfaz de la herramienta de forma similar. Se realizó una encuesta a los alumnos del curso de

Procesamiento de Imágenes de la Fundación Universitaria San Martín de la ciudad de Bogotá en el primer semestre de 2005, en la cual se indagó acerca de la preferencia en la utilización de las herramientas para el desarrollo de sus algoritmos. Los resultados de esta encuesta fueron igualmente favorables para las tres herramientas evaluadas, ya que a la curva de aprendizaje fue similar.

Resultados del procesamiento: en los 20 algoritmos realizados, se obtuvo el mismo resultado en las tres herramientas.

IX. CONCLUSIONES

Con este proyecto se buscaba desarrollar una herramienta computacional de software libre que les permitiera a los programadores de procedimientos de procesamiento de imágenes la construcción de programas de forma gráfica, con el fin de mejorar la productividad en la realización de pruebas y ajustes de los algoritmos.

En el mercado se encontró la herramienta *Khoros* (ahora conocida como *VisiQuest*) que es un software comercial muy completo que incluye una librería con más de 500 operaciones de procesamiento y además tiene un editor gráfico de programas llamado Cantata. Se buscó una librería de procesamiento de imágenes de amplia difusión y que pudiera utilizarse en el desarrollo de software libre. Entre las opciones, se seleccionó la librería Java Advanced Imaging - JAI de Sun Microsystems. Se desarrolló un editor de programas que se conecta directamente a la librería JAI. La herramienta permite conectar los resultados de un proceso a la entrada del siguiente de forma visual.

La herramienta desarrollada permite la creación de condicionales, ciclos y procedimientos en los programas gráficos de procesamiento de imágenes.

Las limitantes que tiene el software están dadas por la funcionalidad que ofrece JAI. En la actualidad, se ofrecen 46 diferentes procesos en la herramienta, soportados directamente en la funcionalidad de JAI. Las pruebas efectuadas al software ofrecieron los mismos resultados de las herramientas comerciales. Este es el resultado principal esperado con este proyecto. La arquitectura del software permite que fácilmente se pueda cambiar de librería de procesamiento de imágenes.

Agradecimientos

Este documento se realizó con la colaboración del Ingeniero Rodrigo Moreno, Director del Programa de Ingeniería de Sistemas de la Fundación Universitaria San Martín.

Referencias

[1] Accusoft Inc. Sitio Web de VisiQuest. http://www.accusoft.com/imaging/visiquest/home.asp. 2005. Navegada en Mayo de 2005.
[2] Barbosa, C. y Morales P. Software para Procesamiento de Imágenes. Fundación Universitaria San Martín. 2005.
[3] CONAE (Comisión Nacional de Actividades Espaciales de Argentina) y AgriSat. Sitio Web de AreaSat. http://www.elsitioagricola.com/Soft/agrisat/areasat1_1.asp. 1999 - 2004. Navegada en Mayo de 2005.
[4] Gilbert, D. y Morgner, T. Sitio Web de la librería JFreeChart. http://www.jfree.org/jfreechart/. 2003-2005. Navegada en Mayo de 2005.
[5] Gonzalez, R y Woods, R. Digital Image Processing, 2nd edition. Prentice Hall. 2002.
[6] Image Processing Toolbox. Matlab – User ´s Guide ver 5, Sitio Web www.mathworks.com. Navegada en Julio de 2005.
[7] Jordán, R. y Lotufo R. Arquitectura del Modelo de datos de Khoros. http://www.csc.fi/visualization/dipcourse/html/k2tools/polymorphic/polymorphic.html. 1995. Navegada en Octubre de 2004.
[8] Rogan, J. Notas del curso Métodos de la Física Matemática II. http://macul.ciencias.uchile.cl/~jrogan/cursos/mfm2p00/. Universidad de Chile. 2000. (Navegada en Mayo de 2005)
[9] Sun Microsystems, Inc. Java Advanced Imaging (JAI) API Documentation. http://java.sun.com/products/java-media/jai/forDevelopers/jai-apidocs/index.html. 2005. Navegada en Mayo de 2005.
[10] Universidad de Hannover, Departamento de Ingeniería Electrónica y Tecnología Informática. Parts of Khoros. http://www.tnt.uni-hannover.de/soft/imgproc/khoros/khoros1/more-info.html. 2000. Navegada en Mayo de 2005.

Segmentación vascular y caracterización de placas ateroscleróticas en imágenes de tomografía computarizada 3D

María Alejandra Zuluaga[1]
ale-zulu@uniandes.edu.co

Sergio Iván Mesa[1]
s-mesa@uniandes.edu.co

Luis Felipe Uriza C.[2]
lfuriza@cable.net.co

Marcela Hernández Hoyos[1]
marc-her@uniandes.edu.co

[1] Universidad de los Andes, Grupo Imagine, Grupo de Ingeniería Biomédica,
Bogotá, Colombia, AA 4976

[2] Hospital Universitario San Ignacio, Departamento de Radiología,
Bogotá, Colombia

Abstract

This paper presents a method for vascular segmentation and atherosclerotic plaque characterization in 3D computed mography images. The first step towards this purpose is the extraction of the vessel axis by an expansible skeleton ethod. It uses an iterative estimation-prediction scheme, multi-scale analysis of image moments, and second-order shape odel. Vessel and plaque boundaries are then detected in the planes locally orthogonal to the centerline. Contour points are :termined using the local maximum of the gradient in the image along the radial directions from a point belonging to the :ssel centerline. Experimental results are presented in 3D diagnostic images of pathological carotid arteries.

eywords: vascular image processing, computed tomography, vessel segmentation, image moments, atherosclerotic plaque ıaracterization.

Resumen

Este artículo presenta un método de segmentación vascular y caracterización de placas ateroscleróticas en imágenes de mografía computarizada 3D. El primer paso hacia este objetivo es la extracción de la línea central de la arteria por medio : un método de esqueleto extensible. Este método utiliza un esquema estimación-predicción iterativo, análisis multi-escala : momentos de la imagen y un modelo de forma de segundo orden. Los contornos vasculares y de placas son detectados en ıa segunda etapa sobre los planos localmente perpendiculares a la línea central. Los puntos de los contornos están :terminados por una búsqueda de los máximos locales del gradiente de intensidad, calculado en direcciones radiales a ırtir del punto del eje central de la arteria. Resultados experimentales son presentados sobre imágenes diagnósticas 3D de terias carótidas patológicas.

ılabras claves: procesamiento de imágenes vasculares, tomografía computarizada, segmentación arterial, momentos de ıagen, caracterización de placas ateroscleróticas.

INTRODUCCIÓN

La arteriosclerosis es una enfermedad degenerativa caracterizada por el estrechamiento progresivo de las arterias. Su :sarrollo se debe a la acumulación de lípidos, células fibrosas, células musculares, cristales de colesterol y calcificaciones ı las paredes arteriales. Estos depósitos forman lo que se conoce como placas ateroscleróticas o de ateroma, las cuales,)co a poco obstruyen las arterias, dando origen a la estenosis arterial. En años recientes, la arteriosclerosis se ha convertido ı un problema de salud pública debido a la alta influencia que tiene en las enfermedades del sistema circulatorio tales)mo infarto cerebral y del miocardio, considerados dentro de las primeras causas de mortalidad en la población adulta undial. La angiografía por rayos X convencional, el ultrasonido doppler, la tomografía computarizada (TC) o escanografía

la angiografía por resonancia magnética (ARM) son las técnicas más comúnmente utilizadas en diagnóstico, planeación el tratamiento y seguimiento de esta patología. Tradicionalmente se ha utilizado el grado de estenosis como única medida la gravedad de la enfermedad arteriosclerótica. Sin embargo, diferentes estudios han demostrado que el nivel de estenosis es suficiente para la caracterización de la patología debido a que cada placa de ateroma y su morfología individual tienen ectos diferentes [14]: las placas con alto contenido lípido y capas fibrosas delgadas son más propensas a la ruptura evando a trombosis y eventos vasculares) que las placas con contenido lípido disminuido y capas de fibra gruesa [4]. Por nsiguiente, la caracterización de la placa (tamaño y morfología) es fundamental para determinar su vulnerabilidad y nstituye hoy en día uno de los mayores retos de investigación en procesamiento de imágenes vasculares.

Este artículo presenta un método de segmentación y caracterización de estructuras vasculares obtenidas por TC. Este étodo combina un proceso de seguimiento de la arteria, basado en el análisis multiescala de momentos de inercia para traer su eje central y una búsqueda de contornos vasculares y de placas en los planos localmente perpendiculares al eje.

El artículo está organizado de la siguiente manera. El numeral 2 presenta una breve descripción de trabajos actuales en gmentación vascular 3D. El numeral 3 presenta los métodos propuestos para extracción de la línea central y de los ntornos vasculares. La sección 4 presenta algunos resultados obtenidos sobre imágenes de arterias patológicas. Una scusión y algunos prospectos de trabajo futuro son finalmente tratados en la sección 5.

ESTADO DEL ARTE

Una completa revisión de métodos de segmentación aplicables a imágenes vasculares 3D es presentado en [19, 27, 41]. a extracción de la línea central de la arteria es esencial para esta tarea, ya que provee información simplificada del vaso ue permite caracterizar la topología del árbol vascular y tomar medidas cuantitativas tales como la longitud curvilínea de la teria, el diámetro o el área de las secciones ortogonales al eje, etc. La línea central (o eje) es el elemento fundamental del odelo de cilindro generalizado usado (algunas veces de manera implícita) en la segmentación de objetos tubulares tales mo las arteras. Este modelo es definido por su eje central y su superficie. Esta última es generalmente aproximada por a pila de contornos planos (2D). Los métodos de segmentación de estructuras vasculares pueden ser clasificados en tres incipales categorías: "contornos primero", "contorno-eje-contorno" y "eje primero", de acuerdo a la manera como definen, traen y acoplan estas dos entidades: línea central y contornos.

1. Enfoque “contornos primero”

La primera categoría reagrupa los métodos que inicialmente extraen los contornos sobre de las imágenes (o cortes) tivas 2D [19, 26, 31]. Después de haber efectuado la discretización de los contornos extraídos, la superficie y su línea ntral son deducidas del conjunto de estos puntos obtenidos. Este enfoque es apropiado en el caso de imágenes cuyas terias son aproximadamente perpendiculares a los cortes nativos. Sin embargo, pueden aparecer errores significativos en caso contrario, cuando la arteria es localmente paralela a los cortes nativos, especialmente cuando los cortes son espesos.

2. Enfoque “contorno-eje-contorno”

La segunda categoría es similar a la primera, ya que el eje es definido por los centros de una serie de contornos 2D. Para nitar los errores de localización de los puntos del eje, estos métodos detectan los contornos en los planos roximadamente perpendiculares al eje. Sin embargo, en esta etapa el eje aun no ha sido extraído. Esta paradoja es lucionada con un algoritmo iterativo propuesto por [38] bajo el nombre de "catéter virtual" y también utilizado por [35] y 1]. El algoritmo completo puede ser descrito de la siguiente manera. Después de una inicialización, típicamente teractiva, del primer punto y de la orientación local del eje: 1) se avanza un paso de acuerdo a la orientación actual, 2) un ntorno es detectado en el plano perpendicular a esta orientación, 3) el centro de este contorno es utilizado para actualizar orientación, enseguida se hace un nuevo paso (regreso a 1). Para detectar los contornos de la arteria, [35] utiliza un odelo de contorno activo mientras que [38] [41] proponen un método que consiste en definir el contorno mediante un mero limitado de puntos situados sobre rayos que parten del eje. Estos puntos están determinados por una búsqueda de s máximos locales del gradiente de intensidad, calculado en la dirección radial con eventuales correcciones para asegurar coherencia con los puntos vecinos [38]. Para la extracción del centro del contorno (que determinará la posición y la ientación del eje), dos métodos ha sido propuestos. En [38] y [35], los puntos del eje son determinados por los centros de avedad de los contornos extraídos. [41] propone una medida de pertenencia al centro de la arteria a partir de las longitudes los rayos lanzados. Es importante anotar que en estos métodos no hay una extracción explícita de los contornos, sino más en una detección de discontinuidades.

Debido al ruido y a las variaciones de contraste, los operadores convencionales de detección de contornos, tales como el áximo del gradiente, pueden producir contornos discontinuos, con lagunas y asperidades. [41] propone un método para tener contornos continuos y lisos apoyándose sobre el formalismo matemático de la búsqueda del camino de costo ínimo. En este método, la búsqueda del contorno es efectuada después de transformación de la imagen original en ordenadas polares, en la cual, un contorno circular es representado por un segmento de recta. El costo de un camino es ntonces determinado por la suma de los gradientes en todos sus puntos.

3. Enfoque "eje primero"

El tercer enfoque consiste en extraer enteramente la línea central de la arteria antes de llevar a cabo la extracción de los ɔntornos. El eje no es, en este caso, definido por los centros de los contornos, sino por otro criterio. Se pueden citar en esta ɪtegoría los métodos de esqueletización [7, 11-13, 23-25, 29, 34,40] los cuales, combinados con métodos de extracción de ɔntornos, pueden constituir un método completo de segmentación. Estos métodos requieren sin embargo una binarización eliminar de la imagen. Entre las técnicas que no requieren binarización preliminar, se destacan la extracción de líneas de esta [2, 30], el método de árbol extensible [1] y el uso de contornos activos geodésicos [5, 6, 21]. Existe también una milia interesante de métodos basados en el análisis de los valores propios de la matriz Hessiana [9, 10, 20, 22, 33, 39, 42, 3]. Estos algoritmos proceden generalmente de la siguiente manera: 1) evalúan en cada punto de la imagen un criterio de ɛrtenencia a una arteria (combinación de valores propios hessianos), a varias escalas; 2) asocian a cada punto de la imagen, respuesta máxima del criterio a través de todas las escalas (el resultado de esta etapa puede ser visto como una nueva ıagen en la cual la intensidad de cada punto indica la probabilidad de que el punto pertenezca a la línea central de una teria; 3) extraen de este mapa de probabilidades los puntos que van a constituir el eje final.

Nosotros hemos escogido el enfoque "eje primero". En este artículo presentamos un método para extraer ıtomáticamente las líneas centrales de arterias, basado en un modelo de esqueleto extensible cuyo crecimiento es ɔntrolado por el análisis multi-escala de los momentos de inercia.

MÉTODO

Siguiendo el principio de las técnicas "eje primero", proponemos un método de segmentación dividido en dos etapas. ɾimero, la extracción de la línea central de la arteria es llevada a cabo usando un modelo de esqueleto extensible [28]. Los ɔntornos de la arteria son enseguida detectados en los planos localmente ortogonales al eje utilizando una combinación de iterios de gradiente e intensidad.

1 Extracción del eje de la arteria

1.1 Descripción del algoritmo

La inicialización del algoritmo de extracción del eje requiere la intervención del usuario para definir un volumen de terés (VOI) que contiene el segmento vascular a analizar y para seleccionar un punto inicial dentro de la arteria. A partir ɔ ese momento, el algoritmo es automático e incluye dos pasos:

Refinamiento (estimación) de la posición del punto actual $\mathbf{x}_i \in \mathbf{R}^3$;

Cálculo de la orientación local de la arteria $\mathbf{e}_i \in \mathbf{R}^3$ y predicción del siguiente (candidato) punto $\hat{\mathbf{x}}_{i+1}$ de acuerdo a esta orientación.

El proceso de extracción es llevado a cabo en las dos direcciones opuestas a partir del punto inicial, y termina cuando los nites del VOI son encontrados. Cada paso está basado en el cálculo de los momentos de la imagen dentro de un sub-ɔlumen esférico llamado *celda de análisis*. La primera versión del algoritmo seguía un enfoque mono-escala en el cual la ɛlda tenía un tamaño fijo predeterminado [15-17]. En esta nueva propuesta se utiliza un enfoque multi-escala para ɛterminar el tamaño de la celda que más se ajusta al diámetro local de la arteria. Actualmente, el algoritmo utiliza dos ɛldas distintas para propósitos diferentes. La primera es usada durante la estimación de la posición del punto actual. Su maño y localización son modificados hasta que la celda esté bien centrada dentro de la arteria y hasta que su tamaño se uste al diámetro local de la arteria. Este proceso es el corazón del enfoque multi-escala y es descrito a continuación. uando la localización y el tamaño de la celda son estables, esta es llamada *celda óptima*. La segunda celda es concéntrica a celda óptima, pero su diámetro es mayor, de forma tal que pueda ser utilizada para estimar la orientación local de la teria. Como esta orientación es utilizada para efectos de la predicción, nosotros llamamos esta esfera *celda de predicción*. continuación se presentan cada uno de los pasos del algoritmo de extracción del eje de forma más detallada.

ɜtimación de la posición del punto actual

De forma similar a la propuesta de [3, 37], se espera que cada punto de la línea central coincida con el centro de avedad de la celda de análisis. Sin embargo, detecciones erróneas pueden ocurrir debido al ruido, bifurcaciones y formas ıtológicas. Con el objetivo de sobrepasar estas limitaciones, se refuerza la continuidad y la suavidad de la línea central por edio de un modelo similar al modelo de "snake". La posición del punto actual (predicho) es refinada bajo la influencia de ıa *fuerza externa* basada en propiedades de la imagen y la reacción de *fuerzas internas* del modelo. La *fuerza externa* atrae punto hacia el centro de gravedad $\mathbf{x}_{i+1}^{\mathrm{G}}$ de la celda centrada en $\hat{\mathbf{x}}_{i+1}$. Su contribución puede ser escrita como el siguiente ɜsplazamiento:

$$\mathbf{d}_{i+1}^{\mathrm{ext}} = -\left(\hat{\mathbf{x}}_{i+1} - \mathbf{x}_{i+1}^{\mathrm{G}}\right). \qquad (1)$$

Las *fuerzas internas* imponen restricciones de forma al modelo, en particular *continuidad* ponderada por un coeficiente $w_c \in \mathbf{R}$ y *suavidad* ponderada por $w_s \in \mathbf{R}$. En nuestros experimentos, utilizamos $w_c = 0.5$ and $w_s = 0.1$. El desplazamiento debido a las fuerzas internas es:

$$\mathbf{d}_{i+1}^{\text{int}} = -w_c\left(\hat{\mathbf{x}}_{i+1} - \mathbf{x}_i\right) - w_s\left(\hat{\mathbf{x}}_{i+1} - 2\mathbf{x}_i + \mathbf{x}_{i-1}\right). \quad (2)$$

De esta manera, el punto actual es trasladado hacia la siguiente posición:

$$\mathbf{x}_{i+1} = \hat{\mathbf{x}}_{i+1} + \mathbf{d}_{i+1}^{\text{ext}} + \mathbf{d}_{i+1}^{\text{int}}. \quad (3)$$

Adicionalmente, el diámetro de la celda de análisis debe ajustarse al diámetro local de la arteria. Sino se hace esto, cuando la celda es demasiado pequeña, su centro de gravedad no tiene significado. O por el contrario, cuando la celda es demasiado grande puede contener fragmentos de estructuras vecinas que pueden modificar su centro de gravedad. Los valores propios de la matriz de inercia son analizados para hacer evolucionar el tamaño de la celda. Cuando la celda es incluida dentro de la arteria, su contenido no tiene orientación privilegiada y los momentos de inercia son idénticos para cualquier eje. En el caso contrario, cuando la celda es lo suficientemente grande para contener una porción de la arteria, hay un único eje, correspondiente a la orientación local de la arteria, alrededor del cual el cilindro giraría con un momento de inercia mínimo. Por consiguiente, la búsqueda del tamaño apropiado de la celda está basada en la detección del límite entre comportamiento esférico y cilíndrico de la estructura contendida dentro de la celda. La adaptación del tamaño de la celda es llevada a cabo "inflando" la celda hasta que los tres valores propios de la matriz de inercia sean aproximadamente iguales entre ellos o "desinflándola" cuando los valores propios son significativamente diferentes.

Cálculo de la orientación local y predicción de un nuevo punto

La orientación local de la arteria es definida por el vector propio $\mathbf{e}_i$ asociado al valor propio más pequeño de la matriz de inercia de la celda centrada en $\mathbf{x}_i$ (el punto actual del eje). Como se mencionó anteriormente, el radio ρ_{pred} de la celda de predicción utilizada para calcular la orientación local debe ser lo suficientemente grande para englobar una porción de la arteria de interés y lo suficientemente pequeño para que la arteria pueda ser considerada como un cilindro recto. Los mejores resultados experimentales fueron obtenidos cuando el diámetro de la celda es entre 1.5 y dos veces el diámetro de la arteria.

La predicción del nuevo punto del eje es llevada a cabo a lo largo del vector propio $\mathbf{e}_i$ con una magnitud fija $\delta \in \mathbf{R}$:

$$\hat{\mathbf{x}}_{i+1} = \mathbf{x}_i + \delta\,\mathbf{e}_i. \quad (4)$$

El parámetro δ debe ser igual a la mitad del radio local de la arteria.

1.2 Manejo de la especificidad de las imágenes de tomografía computarizada

Este método de extracción de líneas centrales fue originalmente concebido para ser aplicado en angiografías por resonancia magnética sustraídas [28]. En dichas imágenes, el lumen arterial es resaltado, mientras que los tejidos estacionarios que lo rodean y en particular las placas de ateroma son removidos por medio de una sustracción digital de dos conjuntos del mismo volumen, adquiridos antes y después de la inyección de un medio de contraste. En este caso, la arteria puede ser modelada como un sólido homogéneo rodeado por un fondo negro (o al menos de densidad casi nula comparada con la densidad de la arteria) y los momentos de inercia pueden ser calculados sobre la imagen original. En el caso de las imágenes de tomografía computarizada, el lumen es igualmente resaltado por un medio de contraste, pero los tejidos que lo rodean no son suprimidos. Pueden entonces aparecer en la imagen estructuras con mayor intensidad que el lumen (calcificaciones de la placa, huesos) o con menor intensidad (tejido adiposo de la placa).

El fundamento teórico del método reside en el cálculo de los centros de gravedad y de las orientaciones de los cilindros que representan segmentos de la arteria incluidos en las celdas esféricas. Está basado en las propiedades mecánicas de los objetos tubulares. Dichos objetos giran alrededor de su eje principal más fácilmente que alrededor de cualquier otro eje, porque su momento de inercia en esta dirección es mínimo. Para aplicar esta teoría a las imágenes 3D se hace una analogía entre las intensidades de voxeles y las masas de partículas de un sólido. En este contexto, los puntos pertenecientes a calcificaciones o a huesos cercanos a la arteria poseen una “masa” mayor que los puntos del lumen y por consiguiente el eje es atraído por estos puntos, resultando en una línea central errónea (Figura 1 a).

(a) (b)

Figura 1. Posibles comportamientos del algoritmo de extracción de eje en presencia de calcificaciones. En gris se representa la calcificación y en rojo el eje extraído. (a) Manteniendo la calcificación (el eje es atraído por la calcificación). (b) Suprimiendo la calcificación (el eje pasa por el centro del lumen).

Debido a este comportamiento, es necesario eliminar todos los componentes de la imagen que no hagan parte del lumen que puedan interferir en la extracción del eje. Se propone entonces una etapa preliminar de preprocesamiento basada en un goritmo de crecimiento de regiones con un criterio compuesto de homogeneidad (doble umbral) a partir de un punto milla al interior del lumen, seleccionado manualmente. Para eliminar las calcificaciones dejando el resto de la arteria tacta, se debe seleccionar un valor de umbral inferior cercano al nivel típico de la sangre. Este valor es fijado teractivamente por el usuario alrededor de 150UH que es el valor mínimo que puede presentar la sangre según el estudio tométrico de imágenes TC de arterias carótidas realizado por Florez [8]. Por otra parte, el umbral superior debe garantizar ıe las calcificaciones sean eliminadas sin que se afecte el lumen. Es por esto que se optó por utilizar como umbral superior ı valor alrededor de 520UH [8]. Al manejar dos umbrales la estructura resultante es aproximadamente cilíndrica con ıecos que representan las calcificaciones eliminadas. Con esto, se pretende que las calcificaciones no sean tenidas en ıenta como parte de la arteria al momento de extraer el eje. Como consecuencia, el eje resultante pasa por el centro del men (Figura 1 b).

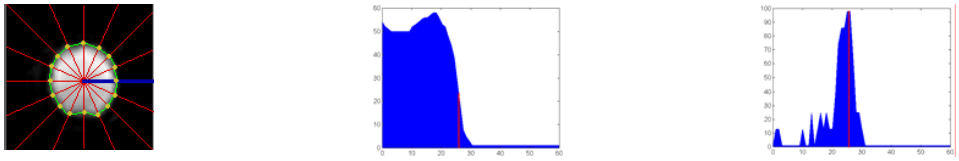
Cuando el eje de la arteria ha sido calculado, procedemos a la detección de los contornos vasculares y de las lcificaciones en los planos localmente perpendiculares al eje, extraídos de la imagen original (sin preprocesamiento).

2 Extracción de los contornos vasculares y de calcificaciones

El método de extracción de contornos vasculares y de calcificaciones que proponemos en este artículo está inspirado en técnica detección de contornos planteada por Wink [41], la cual fue descrita brevemente en la sección 2. Como se encionó anteriormente, en esta etapa no hay una extracción explícita de los contornos, sino más bien una detección de scontinuidades. Por consiguiente, esta detección es seguida de un algoritmo de extracción de contornos bien definidos, nto de la pared vascular como de las calcificaciones por medio de isocontornos.

2.1 Detección de discontinuidades: lumen y calcificaciones

El método propuesto por Wink [41] consiste en definir el contorno por un número limitado de puntos situados sobre yos lanzados a partir del punto del eje. Ya que los píxeles localizados al interior de la arteria presentan una intensidad ayor que los que la rodean, el borde de la arteria se define como el punto en el cual el gradiente en la dirección radial canza el primer máximo por encima de un umbral t (Figura 2 y Figura 3). El valor de este umbral t debe ser gnificativamente mayor que el valor del nivel de ruido del conjunto de datos para evitar inconvenientes.



(a) (b) (c)

Figura 2. Detección del contorno vascular en un plano perpendicular al eje de la arteria, por búsqueda radial del máximo del gradiente (a). Perfil de intensidad (b) a lo largo de un rayo (línea azul) presenta una fuerte ruptura a nivel del borde del lumen vascular. El módulo del gradiente correspondiente (c) presenta un máximo marcado en este punto. La curva (contorno) resulta de una interpolación de los puntos detectados por los diferentes rayos.

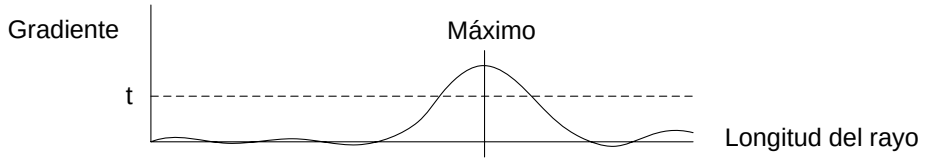


Figura 3. Detección de borde. A lo largo de un rayo (vector) se calcula el gradiente. Si se encuentra un máximo por encima de *t*, se determina que el punto del rayo hace parte del borde de la arteria.

Tal como lo plantea Wink [41], las calcificaciones en las imágenes TC pueden causar que los píxeles cerca de las ıredes arteriales tengan un nivel de intensidad mayor que los píxeles del centro. Como el gradiente se calcula en la rección de los rayos, las calcificaciones son tomadas como parte de la arteria. Si, por el contrario, se utiliza el valor ›soluto del gradiente el rayo, este se detiene en el primer cambio de intensidad que detecta y da una estimación errónea del ıcho de la arteria.

Nuestra propuesta utiliza como base el criterio de gradiente que maneja Wink [41] para detectar el borde de la arteria ·ro introduce algunas variaciones con el fin de poder detectar los componentes de placa que puede contener la arteria.

A partir del punto del eje central de cada plano perpendicular al eje se generan una serie de rayos en dirección radial ıcia el borde de la arteria. Por cada rayo, existe otro rayo apuntando en la dirección opuesta. Para determinar en qué omento se pasa de una región de la arteria a otra (lumen-calcificación o viceversa) o se llega al borde de la arteria ·ansición lumen-fondo o calcificación-fondo), se analiza la magnitud del gradiente de la imagen calculado en la misma rección radial mediante la siguiente expresión:

$$\frac{\partial L}{\partial \vec{r}} = \frac{-\vec{r}\cdot\nabla L}{\left|\vec{r}\right|} \tag{5}$$

Donde *L* es la imagen, *r* el rayo en cuestión y ∇L el gradiente de la imagen.

Para hacer más robusto el algoritmo con respecto a las variaciones de intensidad ocasionadas por el ruido, se efectúa un trado gaussiano antes de calcular el gradiente. Este procedimiento equivale a convolucionar la imagen original *I* con una ·rivada gaussiana ∇G:

$$\nabla L(\vec{x},\sigma) = I(\vec{x}) * \sigma\nabla G(\vec{x},\sigma) \tag{6}$$

El kernel gaussiano de convolución se denota como:

$$G(\vec{x},\sigma) = \frac{e^{\frac{-|\vec{x}|^2}{2\sigma^2}}}{2\sigma^2} \tag{7}$$

onde sigma es la desviación estándar de la curva Gaussiana. Los mejores resultados de detección de bordes son obtenidos ›n un valor de sigma alrededor de 2.5.

A partir de los estudios fotométricos realizados por Florez [8] y los estudios de Ramachandrapa [32] se puede detectar ı patrón de comportamiento en los niveles de intensidad de los componentes de la arteria de la siguiente forma:

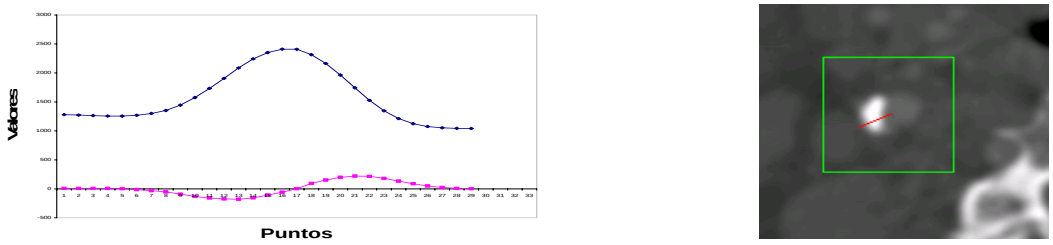
- Las calcificaciones presentan niveles de intensidad de 400 -1000 UH
- Las intensidades asociadas al lumen están en un rango de 150 - 520 UH
- El tejido blando y las zonas hipodensas de la placa tienen intensidades asociadas en un rango de -100 – 100 UH

Aunque de acuerdo a las cifras anteriores, las calcificaciones pueden llegar a tener niveles de intensidad similares a los › lumen, se puede generalizar diciendo que la mayor intensidad siempre la tendrán las calcificaciones, seguidas de lumen y s intensidades más bajas las tendrán el tejido blando y los planos grasos (fondo de la imagen). Teniendo en cuenta estas ıracterísticas y recordando que el gradiente presenta picos cuando hay cambios de intensidad, a partir de la ecuación (6) se ıede definir que (Figura 4):

- Un cambio de zona lumen a calcificación presenta un pico negativo en el gradiente.
- Un cambio de zona lumen a fondo genera un pico positivo en el gradiente.
- Un cambio de zona calcificada a lumen presenta un pico positivo de gradiente.
- Un cambio de zona calcificada a fondo presenta un pico positivo de gradiente.

Con base en este comportamiento, se puede entonces definir que existe un contorno en el punto donde el gradiente cance un máximo o un mínimo local por encima o por debajo de un umbral. Ya que el recorrido siempre parte del interior › la arteria (nunca se parte del fondo de la imagen), la única situación en que se puede presentar un mínimo local es cuando

pasa de lumen a calcificación. Por lo tanto, cada vez que se detecte un mínimo local el contorno puede ser marcado como ı paso de lumen a calcificación.



(a) (b)

Figura 4. Perfiles radiales de gradiente e intensidad al cambiar de zonas en la arteria. (a) En azul se presenta el perfil de ıtensidad y en magenta el perfil de gradiente. Se observa un incremento en el nivel de intensidad (lumen a calcificación) ue se ve reflejado en un mínimo local del gradiente. Luego se presenta una disminución de la intensidad (calcificación a lumen o calcificación a fondo) que se refleja en un máximo local del gradiente. (b) Imagen sobre la cual se hace el análisis. En rojo se presenta el rayo sobre el cual se determinan los perfiles.

Cuando se presenta un máximo local no ocurre lo mismo ya que puede tratarse de tres situaciones: lumen a fondo, lcificación a fondo o calcificación a lumen. Para poder determinar qué tipo de contorno se encuentra al detectar un áximo local, se hace uso de la información fotométrica arrojada por el estudio de Flórez [8] Una vez se detecta el máximo cal, se evalúa el nivel de intensidad en este punto y se compara con los niveles propuestos por Flórez [8] para determinar ıal es la nueva zona. De esta forma el contorno se marca como el paso de lumen a la nueva zona (Figura 5).

El criterio del gradiente por encima de un umbral para detectar los contornos no arroja resultados satisfactorios cuando rayo se encuentra en una zona calcificada. Por lo general, con este criterio la calcificación tiende a sobredimensionarse. ıra evitar este inconveniente, la detección de contornos de la zona calcificada se hace únicamente utilizando información tométrica. Se sabe que la mayor intensidad se encuentra en las zonas calcificadas. Por eso, cuando en el recorrido del yo, estando en una zona calcificada, se detecta que las intensidades empiezan a decrecer se determina que se ha llegado al ıntorno y se marca el contorno como paso de calcificación a nueva zona.

(a) (b)

gura 5. Detección de contornos. (a) Imagen original. (b) Resultado de la detección de contornos. En magenta, el recorrido de los rayos al pasar por lumen. En azul, el recorrido de los rayos al detectar calcificación.

El valor de umbral que se utiliza para determinar el máximo y mínimo local es asignado manualmente al iniciar el goritmo y se utiliza el mismo valor (con cambio de signo) para la detección del máximo y el mínimo. El valor de este nbral está en un rango entre 10 y 20. Algunas veces el pico de gradiente puede ser muy pequeño y no superar el umbral ignado, generando la no detección de ningún contorno. Para evitar este inconveniente el umbral es modificado námicamente cuando se presenta esta situación. Cada vez que se termina el recorrido radial sin que se haya encontrado gún punto que supere el umbral, el umbral se decrementa en una unidad y se vuelve a dar inicio al recorrido desde el ntro. El proceso se repite hasta que se detecte un punto que sobrepase el umbral.

La distancia entre cada uno de los rayos que se lanza es de 5º, lo que significa un total de 72 rayos. La selección de este ırámetro es importante ya que puede mejorar la detección y definición de los contornos. Un número muy bajo de rayos ıede causar la no detección de zonas. Un número muy alto de rayos genera retardo en el procesamiento sin que la formación que se aporte sea importante.

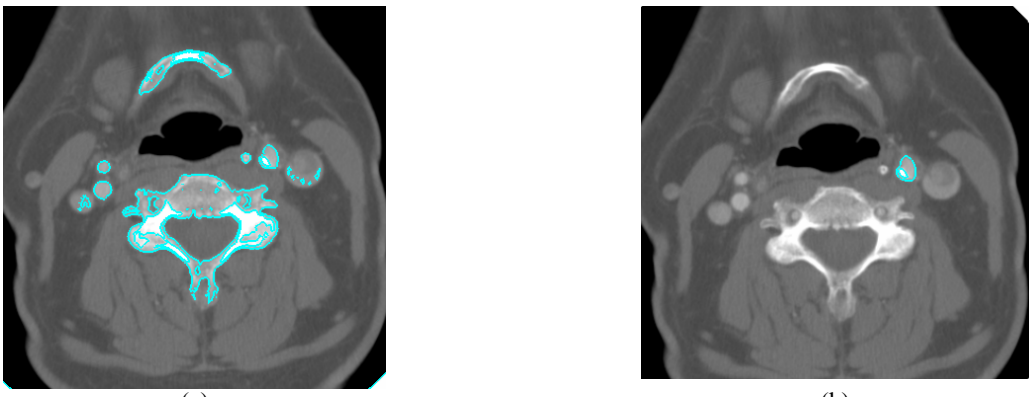
2.2 Extracción de contornos bien definidos por medio de isocontornos

Como resultado del paso anterior, se obtiene una detección de discontinuidades que delinean aproximadamente los ɔrdes del lumen y de las calcificaciones presentes en la imagen. Con el objetivo de obtener contornos bien definidos, esta

:tección es seguida de un algoritmo de extracción de contornos, tanto de la pared vascular como de las calcificaciones por edio de isocontornos.

Un isocontorno [36] divide el espacio de una imagen g en dos subdominios, el exterior, en el cual $g(x,y) < c$ y el interior, ı el cual $g(x,y) > c$, donde c es un umbral denominado isovalor. Bajo esta formulación, es posible considerar las estructuras ı cuestión (pared vascular y calcificaciones) como agrupaciones de píxeles cuya intensidad y distribución espacial permite . delineación mediante isocontornos.

La adecuación de los contornos obtenidos es posible mediante la definición del isovalor como un porcentaje de la tensidad máxima de la estructura de interés, ya sea el lumen o la(s) calcificación(es). Este porcentaje se mantiene ınstante en los cortes pertenecientes a un mismo estudio y su calibración es dependiente de las condiciones de adquisición :l mismo. Se calcula un isovalor por cada calcificación así como para el lumen, que se obtiene al realizar la detección de scontinuidades del paso anterior. Es importante anotar que debido a la similitud de intensidades con otras estructuras no levantes en la imagen, pueden detectarse varios isocontornos para un mismo umbral (Figura 6 a). La supresión de los ıntornos no relevantes es realizada mediante la selección del isocontorno más cercano al centro de gravedad de los puntos ırrespondientes al borde de la estructura (lumen o calcificación) detectados en la etapa preliminar de detección de scontinuidades (Figura 6 b).



(a) (b)

Figura 6. Proceso de generación del isocontorno de la pared vascular. (a) Generación preliminar de todos los socontornos presentes en la imagen con un isovalor correspondiente a un porcentaje de la intensidad máxima del lumen. (b) Supresión de contornos no relevantes.

Como resultado de este proceso obtenemos contornos precisos de las estructuras de interés (Figura 7), que pueden ser ilizados posteriormente para efectuar un análisis cuantitativo de la estenosis carotídea. Este objetivo de cuantificación no ría alcanzable mediante la obtención de un contorno basado exclusivamente en los puntos identificados durante la :tección de discontinuidades.

(a) (b)

Figura 7. Obtención de los contornos correspondientes a la pared arterial y la calcificación. (a) Imagen original. (b) Contornos de la pared vascular y de la calcificación.

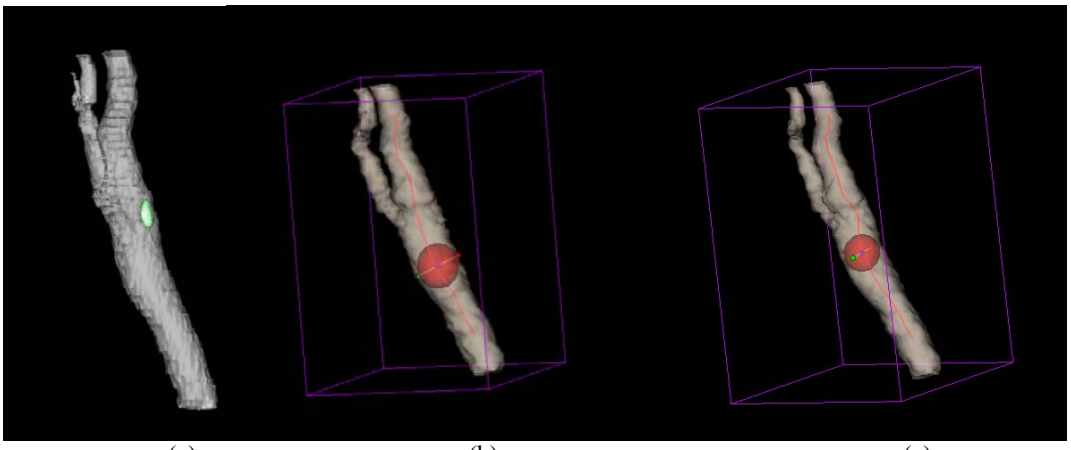
RESULTADOS EXPERIMENTALES

El método de extracción del eje fue evaluado en un trabajo realizado previamente [18] sobre angiografías por resonancia agnética de 35 pacientes (9 mujeres, 26 hombres, con edades entre 47 y 83 años) seleccionados de una base de datos de ıágenes de un estudio multicéntrico llamado CARMEDAS. Se extrajeron los ejes de 140 arterias (carótidas y vertebrales), : los cuales 139 fueron juzgados correctos por dos radiólogos expertos.

En imágenes TC, el algoritmo ha sido aplicado a 18 arterias carótidas (9 pacientes, 4 mujeres y 5 hombres con edades ıtre 52 y 80 años) con resultados visualmente satisfactorios. La figura 8 muestra un resultado sobre una arteria carótida ıtológica. El preprocesamiento de la imagen CT original permite obtener distintos ejes dependiendo de los valores de los

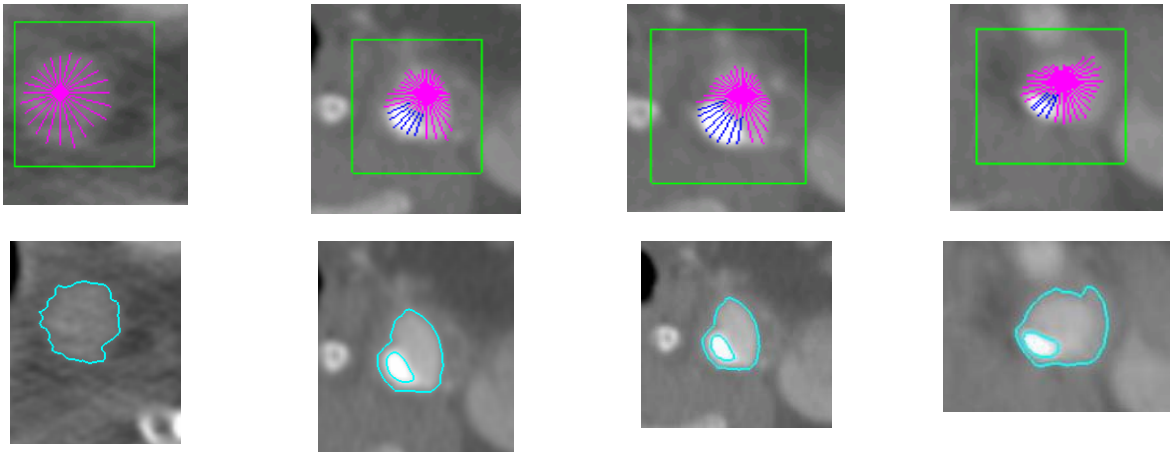
nbrales escogidos para efectuar el crecimiento de regiones. En la Figura 8a. se observa una reconstrucción tridimensional enerada a partir de cortes nativos de una arteria carótida con una calcificación poco después de la bifurcación. Al extraer el e, previa eliminación de la calcificación, el resultado se localiza en el centro del volumen (Figura 8b), en contraste con el e que es atraído por la calcificación debido a su intensidad.

En cuanto a los métodos de detección de discontinuidades y generación de isocontornos, estos han sido experimentados bre 80 cortes CT de arterias carótidas, sanas y patológicas. La figura 9 presenta algunos de estos resultados. En la tección de contornos se puede apreciar que se logra la distinción entre calcificación y lumen (Figura 9 fila superior). Se ede observar que los contornos obtenidos por esta detección de discontinuidades no son del todo correctos. En la lcificación, los rayos tienden a pasarse un poco del borde externo. Adicionalmente, si unimos estos puntos de scontinuidad (puntos extremos de los rayos) mediante segmentos de recta, los contornos que obtenemos son bastante egulares. Como se muestra en la fila inferior de la figura 9, el proceso de reconstrucción de los contornos utilizando ocontornos corrige los problemas de las discontinuidades y la sobreestimación de la placa.



(a) (b) (c)

Figura 8. a) Reconstrucción 3D de arteria patológica. b) Extracción del eje posterior a la eliminación de la calcificación. c) Extracción del eje con calcificación



gura 9. Fila superior: detección de discontinuidades por el método de lanzamiento de rayos. En magenta se presentan los yos sobre lumen y en azul los rayos sobre calcificación. Fila ingerior: reconstrucción del contorno mediante isocontornos otenidos al estipular el isovalor como un porcentaje de la intensidad máxima y seleccionar el contorno mas cercano al ntro de gravedad obtenido a partir de la detección de discontinuidades.

A partir de los resultados obtenidos bajo este esquema, se encontró que el mecanismo resultaba útil para la detección de ntornos de lumen y calcificación solo para situaciones en los que la calcificación limita en uno de sus extremos con la red arterial. Para configuraciones diferentes este enfoque no resulta válido pues confunde los bordes de la calcificación

ɔn los bordes de la pared arterial. De la misma forma, se encontró que cuando la placa contiene zonas hipodensas estas no gran ser detectadas. Las zonas hipodensas presentan niveles de intensidad del mismo orden al que presenta el fondo. Por ta razón cuando un rayo llega a una zona hipodensa asume que se trata de fondo y detiene su recorrido (Figura 10).

(a) (b)

Figura 10. Limitaciones del algoritmo. (a) La presencia de una zona hipodensa en la mitad de la arteria hace que el lgoritmo la confunda con el borde de la arteria y se detenga el recorrido de los rayos. (b) El paso de calcificación a lumen se confunde con el borde de la arteria y se detiene el recorrido de los rayos.

CONCLUSIÓN

Presentamos en este artículo un método de extracción de líneas centrales y contornos de estructuras vasculares aplicado la caracterización de placas ateroscleróticas en imágenes de tomografía computarizada 3D. Es importante anotar que el étodo de extracción del eje es eficiente si las hipótesis de modelaje son satisfechas. En otras palabras, es necesario que la teria presente una densidad homogénea y que el fondo sea aproximadamente negro. Dado que las imágenes TC nativas no ımplen estas condiciones, se propuso una etapa preliminar de preprocesamiento basada en un algoritmo de crecimiento de giones con un criterio compuesto de homogeneidad (doble umbral) a partir de un punto semilla al interior del lumen, con objetivo de eliminar todos los componentes de la imagen que no hacen parte del mismo. Una evaluación visual de las ıágenes de líneas centrales obtenidas después de este preprocesamiento muestra una estimación precisa de la posición y ientación del eje.

En cuanto al método de extracción de contornos, se propuso un método basado en el comportamiento del gradiente ante variación de intensidades y en el conocimiento de las intensidades típicas de los componentes de la arteria. En una imera etapa se detectan discontinuidades de intensidad entre el lumen y las calcificaciones. Posteriormente, con el ɔjetivo de obtener contornos bien definidos, se aplica un algoritmo de extracción de contornos, tanto de la pared vascular ɔmo de las calcificaciones por medio de isocontornos.

Como trabajo a futuro se plantea el desarrollo de un método para la detección de contornos que esté en capacidad de :tectar las zonas hipodensas de la placa con el objetivo de caracterizar la placa con mayor precisión para fines agnósticos. En la actualidad, por la similitud de estas zonas con el fondo de las imágenes, las zonas hipodensas son :tectadas como fondo lo que lleva a una menor precisión en la caracterización.

Dado que el pronóstico de los pacientes depende de una terapia adecuada, determinada por la cuantificación del grado de tenosis en la actualidad, se hace imperativo el desarrollo de métodos de cuantificación que sean objetivos y reproducibles ɔmo el propuesto. Hacia el futuro es importante una herramienta que complemente la caracterización de la estenosis y de placa obtenida en escanografía para el análisis biomecánico que permitirá la exploración de otras variables pronósticas ıra el desenlace clínico de estos pacientes.

gradecimientos

Los resultados y metodologías presentados en este artículo hacen parte de los siguientes proyectos:

- Proyecto Colciencias No. 12040416468 intitulado “Estudio de la hemodinámica a través de una estenosis de la bifurcación de la arteria carótida y su influencia en las observaciones clínicas. Parte II: Integración del análisis de imágenes médicas y hemodinámico para la toma de decisiones en pacientes con enfermedad arterioesclerótica carotídea.”
- Proyecto ECOS-Nord No. C03S02, Cooperación Francia-Colombia: Análisis interactivo y almacenamiento distribuido de imágenes médicas 3D.

eferencias

] Angella F., Lavialle O., Baylou P. A deformable and expansible tree for structure recovery. **In**: ICIP'98 - IEEE International Conference on Image Processing, October 4-7, 1998, Chicago, Illinois. 1998, pp. 5p.

] Basset-Merle A. Reconstruction 3D filaire de l'arbre coronaire en angiographie par rayons X. PhD Thesis. INSA de Lyon, France, 1999. 231 p.

] Boldak, C., Rolland, Y., Toumoulin, C. and Coatrieux, J. L. An improved model-based vessel tracking algorithm with application to Computed Tomography Angiography', *J. Biocybernetics & Biomed. Engineering.* 2003. Vol. 3, pp. 41-64.

] Choudhury, R., Fuster, V., Badimon, J., Fisher, E., Fayad, Z. MRI and Characterization of Atherosclerotic Plaque: Emerging Applications and Molecular Imaging. *Arterioscler Thromb Vasc Biol*. 2002. pp. 1065-1074.

] Cohen L.D., Kimmel R. Global minimum for active contour models: a minimal path approach. International Journal of Computer Vision. 1997, Vol. 24, No. 1, pp. 57-78.

] Deschamps T., Cohen L. Fast extraction of minimal paths in 3D images and applications to virtual endoscopy. Medical Image Analysis. 2001, Vol. 5, pp. 281-299.

] Flasque N, Desvignes M, Constans J-M, Revenu M. Acquisition, segmentation and tracking of the cerebral vascular tree on 3D magnetic resonance angiography images. Medical Image Analysis 2001;5:173-183.

] Florez, L., Étude photo-métrique des images d'AngioScanner (CTA). Reporte interno. CREATIS. 2004.

] Frangi A.F., Niessen W.J., Hoogeveen R.M., *et al.* Model-Based Quantitation of 3-D Magnetic Resonance Angiographic Images. IEEE Transactions on Medical Imaging. 1999, Vol. 18, No. 10, pp. 946-956.

0] Frangi A.F. Three-dimensional model-based analysis of vascular and cardiac images. PhD Thesis, Utrecht University, The Netherlands, 2001. 174 p.

1] Gagvani N., Silver D. Parameter controlled skeletons for 3D visualization. **In**: IEEE Visualization 1997, October 19-24 1997, Phoenix, AZ, USA. 1997.

2] Gagvani N., Silver D. Parameter controlled volume thinning. Graphical Models and Image Processing. 1999, Vol. 61, No. 3, pp. 149-164.

3] Gerig G., Koller T., Szekely G., *et al.* Segmentation and symbolic description of cerebral vessel structure, based on MR angiography volume data. **In**: H.U. Lemke, M.L. Rhodes, C.C. Jaffee, *et al.* Eds. Computer Assisted Radiology, Berlin. Berlin: Springer, 1993, pp. 359-365.

4] Golledge, J., Greenhalgh, R., Davies, A.H. The symptomatic carotid plaque. *Stroke*. 2000, Vol. 31. pp. 774-781.

5] Hernández-Hoyos M., Orkisz M., Roux J.P., *et al.* Inertia-based vessel axis extraction and stenosis quantification in 3D MRA images. **In**: H.U. Lemke, M.W. Vannier, K. Inamura, *et al.* Eds. CARS'99 - Computer Assisted Radiology and Surgery, 23-26 June 1999, Paris. Amsterdam: Elsevier-Verlag, 1999, pp. 189-193.

6] Hernández-Hoyos M., Anwander A., Orkisz M., *et al.* A deformable vessel model with single point initialization for segmentation, quantification and visualization of blood vessels in 3D MRA. **In**: S.L. Delp, A.M. DiGioia, B. Jaramaz Eds. MICCAI'2000 - Medical Image Computing and Computer-Assisted Intervention, October 11-14, 2000, Pittsburgh, PA, USA. Berlin: Springer, 2000, pp. 735-745. (Lecture Notes in Computer Science).

7] Hernández-Hoyos M., Orkisz M., Puech P., *et al.* Computer assisted analysis of 3D MRA images. RadioGraphics. 2002, Vol. 22, No. 2, pp. 421-436.

8] Hernández Hoyos, M., Orłowski, P., Piątkowska-Janko, E., Bogorodzki, P., Orkisz, M. Vascular centerline extraction in 3D MR angiograms to optimize acquisition plane for blood flow measurement by phase contrast MRI. *Aceptado en Computer Assisted Radiology and Surgery (CARS) 2005.* 22-25 June 2005, Berlin.

9] Juhan V., Nazarian B., Malkani K., *et al.* Geometrical modelling of abdominal aortic aneurysms. **In**: J. Troccaz, E. Grimson, R. Mosges Eds. CVRMed-MRCAS'97, March 1997, Grenoble. Berlin: Springer, 1997, pp. 243-252. (Lecture Notes in Computer Science).

0] Krissian K. Traitement multi-échelle: applications à l'imagerie médicale et à la détection tridimensionnelle de vaisseaux. PhD Thesis, INRIA. Université de Nice - Sophia Antipolis, France, 2000. 271 p.

1] Lavialle O., Angella F., Baylou P. Approche min-max pour la recherche d'un chemin optimal. **In**: 17-e colloque GRETSI, 13-17 septembre 1999, Vannes, France. 1999, pp. 689-692.

2] Lorenz C., Carlsen I.-C., Buzug T.M., *et al.* Multi-scale line segmentation with automatic estimation of width, contrast and tangential direction in 2D and 3D medical images. **In**: J. Troccaz, E. Grimson, R. Mosges Eds. CVRMed-MRCAS'97, March 1997, Grenoble. Berlin: Springer, 1997, pp. 233-242. (Lecture Notes in Computer Sciences).

3] Marquez J.A. Analyse et visualisation des structures complexes en 3D : une approche par frontières discrètes. PhD Thesis, Paris, France: Ecole Nationale Supérieure des Télécommunications, 1999. 276 p.

4] Mori K., Hasegawa J.-I., Suenaga Y., *et al.* Automated labeling of bronchial branches in virtual bronchoscopy system. **In**: W.M. Wells, A. Colchester, S. Delp Eds. MICCAI'98 - Medical Image Computing and Computer-Assisted Intervention, October 1998, Cambridge, MA, USA. Berlin: Springer, 1998, pp. 870 - 877. (Lecture notes in computer science).

5] Mori K., Hasegawa J.-I., Suenaga Y., *et al.* Automated anatomical labeling of the bronchial branch and its application to the virtual bronchoscopy system. IEEE Transactions on Medical Imaging. 2000, Vol. 19, No. 2, pp. 103-114.

6] Nazarian B. Reconstruction automatique de structures anatomiques tubulaires à l'aide de surfaces de forme libre. PhD Thesis. Faculté des Sciences de Luminy. Marseille, France: Université de la Méditerranée, 1996. 145 p.

7] Orkisz M., Hernández-Hoyos M. Models for 3D vascular image analysis. Journal of Medical Informatics and Technologies. 2001, Vol. 2 part 1, pp. 13-22.
8] Orkisz M., Hernández-Hoyos M., From inertia matrix to the analysis of vascular pathologies, Int. Conf. Computer Vision and Graphics, Zakopane (PL), Sept. 25-29, 2002, pp. 6-15.
9] Paik D.S., Beaulieu C.F., Jeffrey R.B., *et al.* Automated flight path planning for virtual endoscopy. Medical Physics. 1999, Vol. 25, No. 5, pp. 629-637.
0] Prinet V., Monga O., Ma S. Extraction of vascular network in 3D images. **In**: ICIP'96 - IEEE International Conference on Image Processing, September 16-19, 1996, Lausanne, Switzerland. 1996, pp. 307-310.
1] Puech W., Passail G., Nicolay S., *et al.* Analyse et amélioration de méthodes de reconstruction 3D de l'aorte à partir d'une séquence d'images. **In**: 17-e Colloque GRETSI, September 13-17 1999, Vannes, France. 1999, pp. 1053-1056.
2] Ramachandrapa, D., Karnataka, B., Ambedkar, R. Relevance of cross sectional anatomy in diagnostic and therapeutic measures.Indmedica: India's premier medical portal. Disponible en línea en: http://www.indmedica.com/cyberlec1.cfm?in=yes&clid=12
3] Sato Y., Westin C.-F., Bhalerao A., *et al.* Tissue classification based on 3D local intensity structures for volume rendering. IEEE Transactions on Visualization and Computer Graphics. 2000, Vol. 6, No. 2, pp. 160-180.
4] Sauret V., Goatman K.A., Fleming J.S., *et al.* 3D topology and morphology of branching networks using computed tomography (CT) - Application to the airway tree. **In**: MIUA'99 - Medical Imaging Understanding and Analysis Conference 1999, 19-20 July 1999, Oxford, U.K. 1999.
5] Swift R.D., Ramaswamy K., Higgins W.E. Adaptive axes generation algorithm for 3D tubular structures. **In**: ICIP'97 - IEEE International Conference on Image Processing, 1997, Sta Barbara, CA. 1997, pp. 136-139.
6] Thirion, J.-P., Gourdon, A. The marching lines algorithm : new results and proofs : part 2, INRIA, Sophia Antipolis, Rapport de recherche 1881-2, Avril 1993 1993.
7] Toumoulin, C., Boldak, C., Dillenseger, J.-L., Coatrieux, J.-L. and Rolland, Y., Fast detection and characterization of vessels in very large data sets using geometrical moments, *IEEE Trans. Biomedical Engineering.* 2001. Vol. 48**,** pp. 604-606.
8] Verdonck B., Bloch I., Maître H. Accurate segmentation of blood vessels from 3D medical images. **In**: ICIP'96 - IEEE International Conference on Image Processing, September 16-19, 1996, Lausanne, Switzerland. 1996, pp. 311-314.
9] Wang K.C., Dutton R.W., Taylor C.A. Improving geometric model construction for blood flow modeling. IEEE Engineering in Medicine and Biology. 1999, Vol. 18, No. 6, pp. 33-39.
0] Wilson L.S., Brown S.F., Young J.A., *et al.* Three-dimensional computer models of abdominal aortic aneurysms by knowledge-based segmentation. **In**: H.U. Lemke, M.W. Vannier, K. Inamura, *et al.* Eds. CARS'99 - Computer Assisted Radiology and Surgery, 23-26 June 1999, Paris, France. Amsterdam: Elsevier-Verlag, 1999, pp. 213-217.
1] Wink, O. *Vessel axis determination,* Print Partner Ipskamp, Amsterdam, The Netherlands. 2004.
2] Yim P., Cebral J., Rakesh M., *et al.* Vessel surface reconstruction with a tubular deformable model. IEEE Transactions on Medical Imaging. 2001, Vol. 20, No. 12, pp. 1411-1421.
3] Young S., Pekar V., Weese J. Vessel segmentation for visualization of MRA with blood pool contrast agent. **In**: W.J. Niessen, M.A. Viergever Eds. MICCAI'2001 - Medical Image Computing and Computer-Assisted Intervention, October 14-17, 2001, Utrecht, The Netherlands. Berlin: Springer, 2001, pp. 491-498. (Lecture Notes in Computer Science).

Uso de técnicas de visualización 3D para la enseñanza de anatomía y fisiología del riñón humano

Oscar Ariza
Universidad de los Andes, Facultad de Ingeniería
Bogotá, Colombia
o-ariza@uniandes.edu.co

Pablo Figueroa Ph.D.
Universidad de los Andes, Facultad de Ingeniería
Bogotá, Colombia
pfiguero@uniandes.edu.co

Gustavo Valbuena M.D., Ph.D.
Universidad de los Andes, Facultad de Medicina
Bogotá, Colombia
guvalbue@uniandes.edu.co

Gabriel Martínez
Universidad de los Andes, Facultad de Artes y Humanidades
Bogotá, Colombia
ga-mart@uniandes.edu.co

Santiago Leal
Universidad de los Andes, Facultad de Artes y Humanidades
Bogotá, Colombia
s-leal@uniandes.edu.co

Abstract

This article presents new techniques to support the process of learning human anatomy based on 3D visualization techniques. A set of tools that visualize parts of organs and their context, at different levels of detail are shown. The main objective is to offer new pedagogical tools for learning human anatomy, physiology, histology and pathology in an integrated manner, extending the current methods of medical illustration. We present a prototype of atlas about the kidney, which shows the concepts in 3D visualization techniques we propose. Such atlas is being designed to be used in courses of the medicine undergraduate program in our University.

Keywords: 3D Atlas of Human Kidney, 3D Educational Material, Medical Illustration, 3D Visualization in Medicine

Resumen

Este artículo presenta nuevas técnicas para apoyar el aprendizaje de la anatomía humana, basadas en técnicas de visualización 3D. Exponemos un conjunto de herramientas que permiten visualizar partes de órganos y su contexto, a distintos niveles de detalle. El objetivo es ofrecer nuevas herramientas pedagógicas al aprendizaje de la anatomía, fisiología, histología y patología humanas de una manera integrada, extendiendo los métodos actuales de ilustración médica. Mostramos los esfuerzos que estamos realizando alrededor de un atlas de información acerca del riñón, el cual está siendo diseñado para ser usado en cursos de nuestra facultad de Medicina.

Palabras claves: Atlas 3D del riñón humano, Material educativo 3D, Modelo 3D del riñón, Ilustración médica

1. INTRODUCCIÓN

Durante siglos se ha usado la ilustración como una herramienta para el proceso de aprendizaje en Medicina. Las ilustraciones han permitido describir de una manera esquemática los hallazgos sobre el cuerpo humano y centrar la atención de estudiantes en los objetos de aprendizaje, sin la complejidad ni los problemas relacionados con la observación directa. Por ser el cuerpo una estructura tridimensional, es interesante ver cómo las técnicas de visualización novedosas pueden ser usadas en el área de ilustración médica. Este proyecto tiene como objetivo mejorar los elementos disponibles para un médico en su proceso de aprendizaje de la anatomía, fisiología, histología y patología del cuerpo humano, con base en nuevas técnicas de visualización tridimensional y a la disponibilidad de ambientes gráficos de alto desempeño, como son los computadores con tarjetas aceleradoras para juegos.

El material existente para estudio del cuerpo humano es principalmente 2D: fotografías, microscopías, cortes (sagital, axial, etc.), los cuales no muestran información de profundidad ni de referencia o contexto, y por lo tanto dificultan la comprensión de la funcionalidad o de la forma tridimensional de lo observado. Creemos que el uso de herramientas 3D puede facilitar el entendimiento de estructuras complejas. Concentramos este estudio en el desarrollo de un medio interactivo acerca del riñón, especialmente en su estructura microscópica. Específicamente, fueron definidos los siguientes propósitos:

- Implementar un ambiente 3D que permita visualizar la estructura del riñón humano.
- Desarrollar mecanismos que permitan explorar y visualizar la estructura del riñón a distintos niveles de detalle, desde la visión macroscópica a la microscópica.
- Crear un banco de animaciones de los principales procesos fisiológicos del riñón humano.
- Crear mecanismos que permitan relacionar y/o comparar los contenidos educativos 2D tradicionales con los presentados por la aplicación de manera tridimensional.
- Desarrollar módulos relacionados con la visualización de procesos patológicos.

Se busca que los usuarios o estudiantes de la aplicación objetivo de este proyecto puedan demostrar conocimientos, habilidades, destrezas y/o actitudes con respecto al reconocimiento de la estructura macroscópica y microscópica del riñón, los procesos fisiológicos del riñón a nivel microscópico y molecular y los procesos patológicos del riñón a nivel funcional.

El artículo consta de un análisis de los antecedentes en el tema, el cual está basado en los distintos tipos de ilustraciones y de material disponible relacionado con la enseñanza de la anatomía y fisiología del riñón humano; luego serán presentadas algunas de las técnicas implementadas para cumplir con los objetivos propuestos junto con unos resultados preliminares y por último presentamos las conclusiones de este trabajo.

2. ANTECEDENTES

Existen en el mercado ambientes que presentan la anatomía y fisiología del riñón humano mediante animaciones 2D, también hay sistemas inmersivos que permiten visualizar órganos u ofrecen mecanismos de navegación que facilitan la búsqueda de anomalías y mejoran los diagnósticos [1], pero que en su gran mayoría no soportan procesos de enseñanza/aprendizaje.

En general, las aplicaciones médicas contemporáneas están relacionadas con procesos de reconstrucción 3D del riñón humano o animal a nivel macro, pues la elaboración de técnicas para reconstruir tejidos o estructuras a nivel micro como el nefrón o el glomérulo se encuentra en su etapa inicial [2][3][4]. La mayoría de las reconstrucciones existentes del riñón, resultado de técnicas de captura como angiografía, IRM[1] o TAC[2], están orientadas a la creación de información volumétrica como apoyo a la interpretación y al diagnóstico, visualizando la estructura de interés sin ofrecer alternativas de simulación o exploración interactiva. En términos generales el material encontrado se puede clasificar de la siguiente forma: Colecciones de imágenes de apoyo al diagnóstico, proyectos de simulación del riñón, modelos 3D estáticos y aplicaciones interactivas para internet. Los siguientes párrafos describen estos tipos de material. Las colecciones de imágenes de apoyo al diagnóstico componen atlas para el estudio de la anatomía del riñón, dichas

[1] Imágenes de resonancia magnética

[2] Tomografías computarizadas.

colecciones en su mayoría son realizadas por artistas del dibujo 2D y 3D. Pueden estar compuestas por fotografías de modelos reales a escala (cerámica, materiales plásticos, etc.) de las partes anatómicas del riñón [20], las cuales son ordenadas temáticamente con el fin de complementar la información ofrecida en algunos libros o sitios web.

Algunas fotografías provienen de procesos de corrosión sobre muestras de tejido o tratamientos con siliconas con el fin de recuperar y visualizar el aspecto tridimensional de estructuras internas como el nefrón o el glomérulo [5][6], pero en la mayoría de los casos los conjuntos de fotografías son ofrecidos como material de apoyo para la enseñanza de interpretación o diagnóstico médico [21], sin ofrecer información tridimensional que facilite la comprensión de las estructuras que componen el riñón, sus relaciones espaciales o de proporcionalidad. Existen conjuntos de imágenes que son presentadas a manera de tutorial o laboratorios virtuales para mostrar partes anatómicas a distintos niveles de detalle y/o el proceso de filtrado realizado por el riñón [19]. Hay ilustraciones que agregan información acerca de profundidad [21], pero no conservan las relaciones de escala ni las proporciones existentes en las partes anatómicas reales.

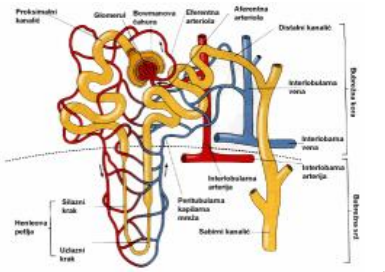


Figura 1. Esquema del nefrón. Fuente: http://hpd.botanic.hr/kem/bkl/pavle/materia/images/nephron.gif

El "proyecto del humano visible" [10], un conjunto de datos para un hombre y para una mujer que representan aproximadamente 17 GB de información, contiene pilas de fotografías reales, imágenes de resonancia magnética y tomografías computarizadas del cuerpo humano completo a una resolución del orden de 1/3 de milímetro, las cuales permiten realizar reconstrucciones del riñón a nivel macro con el fin de hacer visualizaciones de información volumétrica. Existen en internet aplicaciones relacionadas [11][12] que permiten interactuar con las pilas de imágenes para visualizar o hacer cortes sobre las áreas que contienen los riñones y poder observar las relaciones espaciales y de contexto que tienen con respecto a otros órganos que componen el cuerpo humano [30]. Sin embargo, los parámetros de adquisición y la resolución utilizados para obtener estas imágenes no permiten que sean procesadas para obtener información de las estructuras microscópicas contenidas cuyas dimensiones están en el orden de micras, y por consiguiente, dificultan la reconstrucción de tejidos u órganos como el nefrón o el glomérulo.

Los modelos de simulación buscan integrar toda la información biológica funcional y estructural en una arquitectura que permita simular el comportamiento del riñón mamífero en condiciones médicas normales, patológicas o de mutación genética, entre otras. [7] Dicha arquitectura contiene un módulo de simulación que define un modelo matemático [8] del comportamiento de las unidades funcionales del riñón en respuesta a diferentes estímulos o compuestos químicos resultado del funcionamiento metabólico.

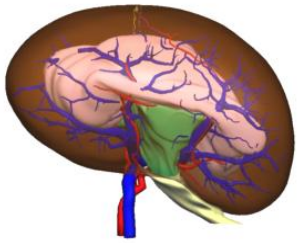


Figura 2. Kidney Simulation Project. Fuente: - http://www.dis.unimelb.edu.au/staff/andrewl/KidneySim.htm

Los modelos 3D del riñón [9] muestran información estática resultado de reconstrucciones de angiografías o geometría creada por artistas en aplicaciones para modelado 3D, y por lo tanto no pueden ser manipulados para brindar

interactividad o producir simulaciones. Existen algunos sitios web comerciales que ofrecen modelos 3D del riñón y de sus principales partes funcionales a diferentes niveles de complejidad [31][32].

Las aplicaciones interactivas para internet representan gran parte del material disponible sobre el funcionamiento del riñón con fines pedagógicos o ilustrativos [13][14][15][16]. Este tipo de material esta desarrollado en su mayoría como animaciones 2D y no contiene modelos tridimensionales ni simulaciones basadas en modelos formales. Existen además problemas con la comprensión de escalas y medidas, ya que las animaciones se presentan sin relación de tamaño ni de contexto. Este tipo de material presenta animaciones de procesos fisiológicos como intercambio, transporte de fluidos, ósmosis y procesos metabólicos en relación con otros órganos del cuerpo humano como el hígado, el corazón y los pulmones. Algunas de las aplicaciones web encontradas complementan su contenido con hipertexto y evaluaciones con retroalimentación en línea [17]. Aunque estas herramientas y el material mencionados presentan una funcionalidad interesante para el estudiante, son inherentemente 2D y por lo tanto limitadas para comprender una estructura tridimensional.

3. CONTENIDO

3.1 Análisis pedagógico

El proyecto sigue una metodología basada en las propuestas presentadas en [22][23][24][25] para desarrollos de materiales educativos computarizados (MECs), teniendo en cuenta el papel que desempeñan profesores y estudiantes en el control del aprendizaje y enfatizando el papel del MEC como facilitador en el proceso de aprendizaje. En dicha metodología se generan dos posibles escenarios: uno en que al aprendizaje es dirigido por el profesor y otro en que el aprendizaje es autodirigido (por el mismo estudiante). Se tomó en cuenta dos enfoques para el MEC: un enfoque algorítmico que busca facilitar el aprendizaje por medio de tutoriales en los que se presentan secuencias predefinidas y/o controlables de contenidos 3D, y un enfoque heurístico que busca proveer ambientes con contenido 3D ricos en situaciones que el estudiante debe explorar conjeturalmente para llegar al conocimiento a partir de la experiencia.

Para el desarrollo del proyecto se definieron los siguientes roles con sus correspondientes responsabilidades, las cuales se complementan para cumplir con los requerimientos pedagógicos y técnicos del MEC planteado: Un médico, un asesor, un pedagogo, un evaluador, dos diseñadores gráficos y un desarrollador.

El análisis pedagógico realizado, toma en cuenta las relaciones existentes entre los intervinientes (profesores, estudiantes y contenido) con el fin de establecer los requerimientos relacionados con la población objetivo, los contenidos, los problemas educativos a resolver y los requerimientos técnicos. Dicho análisis esta orientado a solucionar los problemas planteados con las necesidades educativas encontradas con respecto a los estudiantes, el profesor, los materiales utilizados, el tiempo dedicado al estudio del tema y la metodología utilizada en la enseñanza.

3.2 Contenido del material educativo

El contenido del MEC esta compuesto por modelos 3D de las distintas partes anatómicas del riñón humano, animaciones de proceso fisiológicos y patológicos relacionados, imágenes 2D de microscopía de tejidos relacionados e hipertexto asociado. A continuación se presentan algunos conceptos relacionados con las estructuras de datos y las herramientas que van a facilitar que el estudiante interactúe con dicho material adecuadamente con el fin de cumplir con las necesidades educativas planteadas.

Una de las primeras necesidades encontradas fue la creación del contenido tridimensional. La primera opción es el uso de reconstrucciones 3D del riñón humano, pero su adquisición o creación es difícil debido a que las técnicas de reconstrucción al nivel de detalle necesario se encuentran en una etapa inicial de desarrollo como vimos en la sección 2. Por otro lado, aunque comercialmente pueden ser adquiridos modelos 3D de algunas de las partes funcionales del riñón, estos no son apropiados por su costo o por las necesidades del proyecto, las cuales están orientadas a ofrecer contenido 3D esquemático y claro pero que tenga el detalle suficiente que requiere la enseñanza de temas tan complejos como la fisiología renal.

En consecuencia, se inicio la creación de los modelos 3D mas representativos del riñón en los niveles de detalle que contienen información fisiológica o patológica de interés conforme a los objetivos del proyecto, es decir: el riñón a nivel macro, las pirámides renales, el nefrón y el glomérulo inicialmente (ver figura 3).

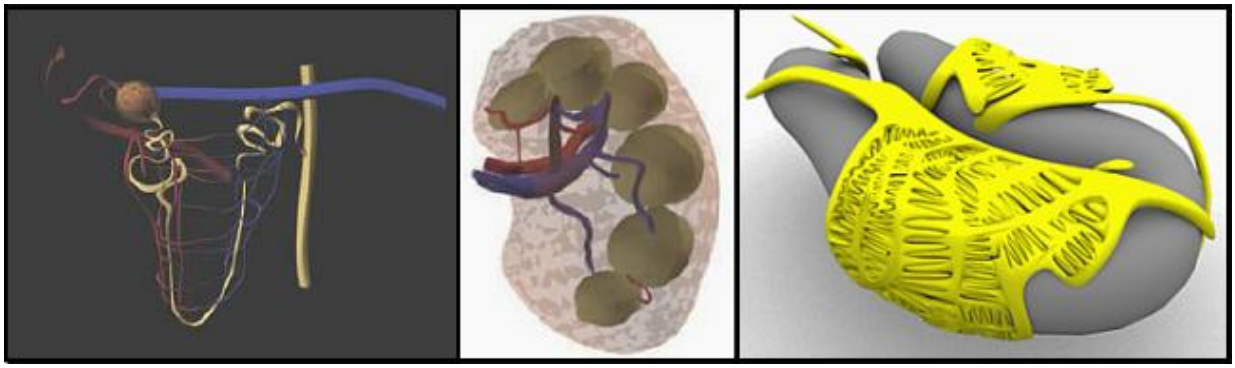


Figura 3. Modelos 3D creados para el proyecto: a la izquierda el nefrón, en el centro el riñón y a la derecha una sección tubular del glomérulo.

Debido a que el riñón es un filtro masivamente paralelo de la sangre, la mayoría de los esfuerzos de modelamiento y procesamiento de geometría 3D están focalizados en la estructura del glomérulo, el órgano principal en el filtrado y absorción de la sangre a nivel molecular. Los túbulos o capilares que conforman el glomérulo así como otras células como los podocitos y tejidos endoteliales son encargados de extractar orina de la sangre, en procesos que se quieren ilustrar en este medio. Es a este nivel que se encuentran las mayores deficiencias en los MECs consultados, y no se encuentra por lo tanto buenas ilustraciones que complementen las imágenes de microscopia existentes.

3.3 Uso de técnicas de foco/contexto

El esquema mostrado en la figura 4 presenta las relaciones nivel de detalle y complejidad que se buscan presentar en el proyecto con respecto a los modelos 3D. La imagen 4.A muestra una microscopía SEM de un glomérulo con 6000 aumentos, la imagen 4.B muestra una aproximación 3D correspondiente a la sección tubular del glomérulo presentada en la imagen 4.A. Se busca que la correspondencia que exista entre un corte transversal realizado sobre el modelo 3D mostrado en la imagen 4.B, realizado por una de las herramientas de corte provistas por la aplicación desarrollada, corresponda de una manera apropiada con la microscopía TEM de uno de los capilares del glomérulo y su podocito adyacente (12000 aumentos) mostrados en la imagen 4.C.

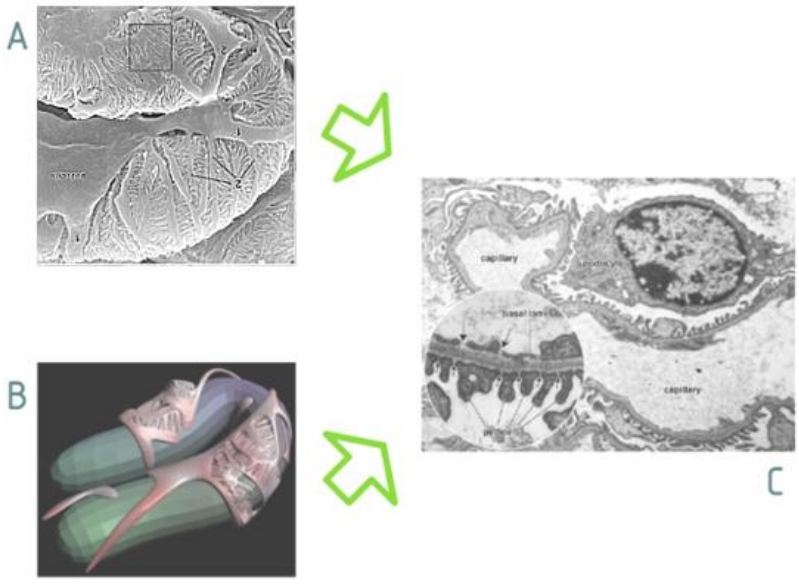


Figura 4. Las imágenes de microscopía fueron tomadas del Atlas de Histología de .H. Ross [36].

Las herramientas presentadas al usuario están relacionadas con técnicas de visualización utilizadas para mostrar el objeto de interés (órgano, parte anatómica, etc.). Debido a la complejidad de dichos objetos de interés (cientos de

nefrones, o unidades funcionales del riñón con sus correspondientes estructuras celulares) es necesario presentar mecanismos de foco-contexto [27][28] que permitan explorar dichos objetos a diferentes escalas o niveles de detalle sin perder la información de contexto y permitiendo navegar por cada una de ellas.

Dichas técnicas a su vez están soportadas por una estructura de datos que almacena las relaciones que existen entre varios órganos y la información asociada a cada uno de estos: geometría, texto asociado, videos o animaciones, etc.. Las relaciones definidas entre los órganos o partes anatómicas tridimensionales son:

- *Contenencia:* Esta relación representa las partes anatómicas contenidas por otra, todas representadas en la misma escala.
- *Nivel de detalle:* Esta relación representa las partes anatómicas contenidas por otra, observadas a una escala diferente a la usada para representar a la parte contenedora.

La figura 5 muestra como ejemplo y de forma simplificada las relaciones de contenencia y nivel de detalle que se pueden definir en el sistema urinario. Inicialmente el sistema urinario (A) puede tener relaciones de contenencia con partes anatómicas como las arterias y venas renales (B), la uretra (D), y las pirámides renales (C). Estas últimas tienen una relación de contenencia con formaciones de tejido que pueden contener cientos de nefrones (E), las cuales tienen una relación de contenencia con tubos colectores o ramificaciones de capilares (F) y una relación de nivel de detalle con el nefrón (G). El nefrón a su vez mantiene una relación de nivel de detalle con el glomérulo (H). Por último el mismo glomérulo tiene tres relaciones de nivel de detalle con partes que contiene pero que deben apreciarse a una escala mucho mayor (10000 aumentos aproximadamente) para poder analizar su estructura y comportamiento fisiológico. La misma figura muestra esquemáticamente la estructura XML que utilizamos para almacenar la estructura de datos.

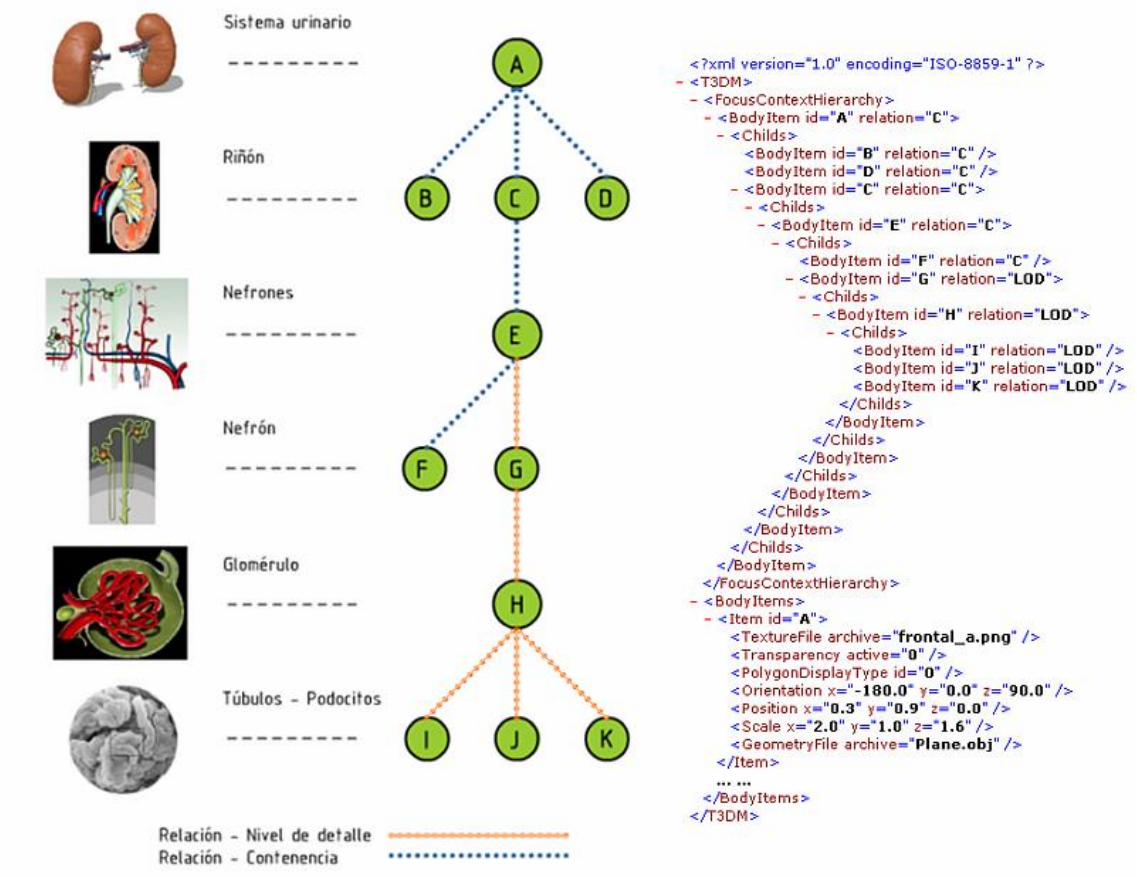


Figura 5. Relaciones de nivel de detalle y contenencia entre las partes anatómicas del riñón humano. Imágenes 3D tomadas de http://www.med-ars.it

El uso de esta estructura de datos permite al usuario navegar libremente por las relaciones, visualizando secuencialmente las partes y cambiando de escala de manera adecuada para mantener las relaciones espaciales y de contexto [26]. Adicionalmente, facilita la manipulación de la geometría asociada (rotación, traslación, etc.), el despliegue de la información multimedia asociada y la ejecución de las herramientas (ver siguiente sección) asociadas a cada uno de los nodos.

3.4 Herramientas y Detalles de implementación

Las herramientas hasta ahora implementadas facilitan la visualización de órganos o partes anatómicas del riñón desde puntos de vista arbitrarios por medio de la realización de cortes sobre la geometría de la parte anatómica de interés. Dichos cortes son realizados por medio de un plano en el ambiente 3D, el cual es manipulado por el usuario para definir una orientación arbitraria y la profundidad del corte. Para manipular el plano de corte en la aplicación debe ser seleccionada la herramienta de corte mediante un menú asociado que además ofrece herramientas para manipular el objeto de interés o cambiar el tipo de cámara empleado, a continuación, el usuario debe realizar una operación de dragging sobre la pantalla para cambiar la orientación del plano de corte, la cual es tangente a una esfera cuyo centroide coincide con el centroide del objeto, o el usario puede mantener presionado un botón para hacer que el dragging modifique el radio de dicha esfera, cambiando de esta forma la profundidad del corte con respecto a la orientación del mismo plano.

Adicionalmente, el usuario puede manipular la orientación de la geometría involucrada y la configuración de las cámaras utilizadas para su visualización (ver figura 6) de tal forma que pueden ser activadas cámaras tipo primera persona, cámaras ancladas a los cambios de posición u orientación del objeto de interés, cámaras para seguir el objeto de interés y cámaras en posiciones predeterminadas que facilitan la visualización de las partes anatómicas. El ambiente de interacción esta implementado para computadores personales (PC) y una implementación para ambientes proyectivos tipo Geowall [37] se encuentra en versión beta.

El modelamiento 3D de las partes anatómicas del riñón y sus estructuras internas fue realizado con Maya [38], las dificultades encontrados están en su mayoría relacionadas con la creación de estructuras complejas con apariencia orgánica, para solucionar este tipo de problemas, fue necesario emplear un tiempo considerable aplicando operaciones de deformación, extrusión, subdivisión y ajuste en modelos aproximados con superficies paramétricas, hasta alcanzar resultados óptimos con respecto a los requerimientos de calidad y detalle.

Una de las operaciones útiles en el proceso de aprendizaje es el corte de las estructuras 3D. Para visualizar estos cortes, utilizamos técnicas que combinan el uso de texturas 3D procedimentales, el uso de arreglos de vértices para acelerar el despliegue de geometría tridimensional y operaciones con plantillas (stencil buffer) sobre los buffers de profundidad y color; aunque se tiene previsto utilizar librerías [35] que permiten ejecutar operaciones de geometría sólida constructiva sobre las representaciones 3D de las partes del riñón. Lo anterior fue implementado por medio de extensiones de OpenGL con el fin de optimizar y mejorar el desempeño de la aplicación.

Fue necesario realizar bastantes pruebas de la herramienta de corte, para soportar todos las posibles configuraciones geométricas presentes en los modelos 3D del riñón humano, es decir, tejidos sólidos contenidos o envueltos por otros tejidos sólidos, estructuras huecas que contienen estructuras o tejidos huecos, células invadidas por otras, etc.

En resumen, la mayoría de los problemas encontrados están en su mayoría relacionados con la creación de la geometría, con la identificación y discriminación de los tejidos durante las tareas de modelado 3d, con el soporte adecuado a las tareas de aprendizaje requeridas y con el trabajo de encontrar el balance entre una ilustración demasiado esquemática que no refleje la realidad y una ilustración demasiado detallada que haga la interacción con el estudiante demasiado lenta.

Actualmente la aplicación ha sido desarrollada bajo los siguientes requerimientos mínimos/deseables: Procesador Pentium IV o superior, 512 Mb. de memoria RAM, acelerador gráfico con una implementación de OpenGL en hardware, teclado y mouse convencionales, pantalla (CRT o FLT) con una resolución mínima de 1024x768 píxeles. Algunos de los resultados que se pueden obtener interactivamente al aplicar la herramienta de corte sobre una sección tubular de un glomérulo se muestran en la figura 6:

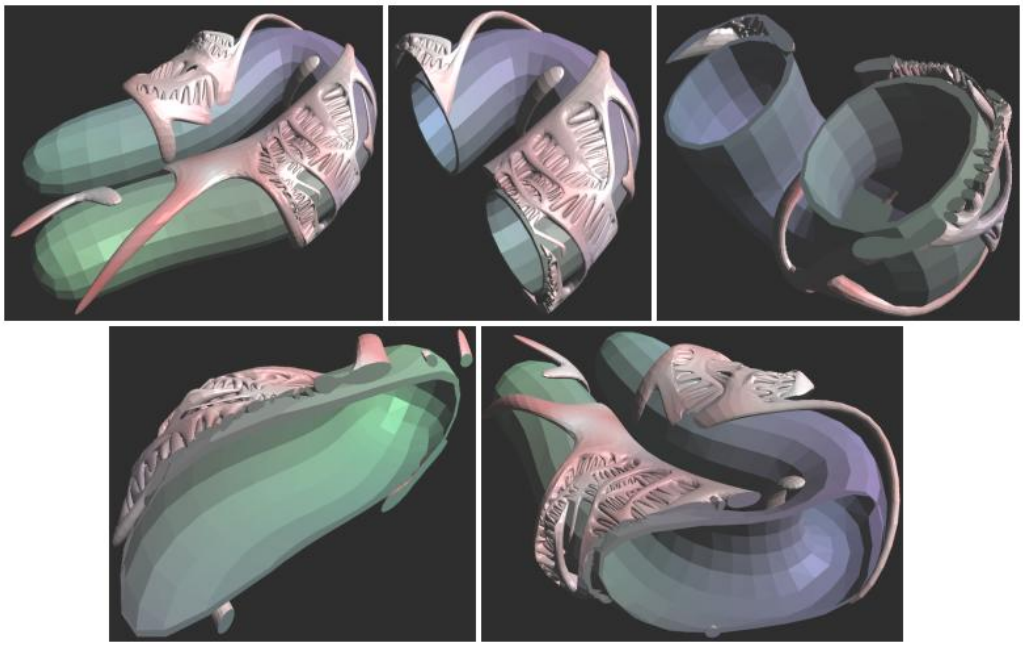


Figura 6. Imágenes de los resultados obtenidos al usar la herramienta de corte sobre secciones del glomérulo.

4. CONCLUSIONES

La ilustración médica puede beneficiarse del uso de técnicas de visualización e interacción 3D. Hemos presentado los avances alcanzados en un proyecto que busca ofrecer material educativo con contenido tridimensional a los procesos de enseñanza y aprendizaje de la anatomía y fisiología del riñón humano. El análisis educativo realizado permitió detectar falencias en el contenido educativo utilizado y facilitó el diseño de herramientas que permiten visualizar y manipular las estructuras del riñón más complicadas para profesores y estudiantes.

Hemos desarrollado una primera versión de un atlas 3D anatómico del riñón, el cual permite hacer cortes arbitrarios a varios niveles de detalle. Dicha aplicación permite también establecer relaciones espaciales y de contexto con las imágenes de microscopía usadas tradicionalmente en la enseñanza de la anatomía del riñón humano. Como trabajo futuro pensamos integrar más herramientas de visualización a nuestro ambiente prototipo y evaluar su uso en condiciones controladas con estudiantes de la facultad de Medicina.

Referencias

[1] Center for Advanced Visualization and Interaction (CAVI). www.systematic.dk. Fecha de consulta: Mayo de 2005.

[2] Thomas L. Pannabecker and William H. Dantzler. Three-dimensional lateral and vertical relationships of inner medullary loops of henle and collecting ducts. Department of Physiology, University of Arizona. *Renal Physiol* 287: F767–F774, 2004.

[3] Xiao Yue Zhai, Henrik Birn, Knud B. Jensen, Jesper S. Thomsen, Arne Andreasen, and Erik I. Christensen. Digital Three-Dimensional Reconstruction and Ultrastructure of the Mouse Proximal Tubule. Department of Cell Biology and †Department of Neurobiology, Institute of Anatomy, University of Aarhus, Aarhus, Denmark. J Am *Soc Nephrol* 14: 611–619, 2003.

[4] Thomas L. Pannabecker, Diane E. Abbott, and William H. Dantzler. Three-dimensional functional reconstruction of inner medullary thin limbs of Henle's loop.Department of Physiology, College of Medicine, University of Arizona, Tucson, Arizona. 25 September 2003.

[5] Simoni Sangiorgi, Alessandro Manelli, Marina Protasoni, Marcela Reguzzoni, Terenzio Congiu and Mario Raspanti.Structure and ultrastructure of microvessels in the kidney seen by the corrosion casting method. Laboratory of Human Morphology – L. Cattaneo – University of Insubria – Varesc, 2004.

[6] Özcan Oguz, Fahri Dere, Ahmet H. Yücel and Behice Durgun-Yücel. Examination of microvascular estructures of midcortal region in sheepkidneys: A Three dimensional approach. Department of anatomy, School of medicine, The Cukurova University, Adana-Turkey, 1991.

[7] The Kidney Simulation Project. http://www.dis.unimelb.edu.au/staff/andrewl/KidneySim.htm. Fecha de consulta: Mayo de 2005.
[8] KSIM: A Kidney Modeling Demo - The Urine Concentrating Mechanism. http://www.necker.fr/kidneysim. Fecha de consulta: Mayo de 2005.
[9] Kidney and nephron medical 3d illustrations. http://www.med-ars.it/galleries. Fecha de consulta: Mayo de 2005
[10] United States – National Library of Medicine - The Visible Human Project. http://www.nlm.nih.gov/research/visible/. Fecha de consulta: Mayo de 2005
[11] The Visible Human Project - data viewer. http://www.dhpc.adelaide.edu.au/projects/vishuman2/ Fecha de consulta: Mayo de 2005.
[12] The Visible Human Project - data viewer. http://anatquest.nlm.nih.gov/AnatQuest/AwtCsViewer/aq-cutaway.html. Fecha de consulta: Mayo de 2005
[13] Kidney patient guide – How kidneys work. http://www.kidneypatientguide.org.uk/site/HKWanim.php. Fecha de consulta: Mayo de 2005
[14] Stephen Z. Fadem, M.D., FACP. The nephron information center - The early renal guidebook - How The Kidney Works. http://www.nephron.com/htkw.html. Fecha de consulta: Mayo de 2005
[15] Kscience – The kidney. http://www.kscience.co.uk/animations/anim_8.htm. Fecha de consulta: Mayo de 2005
[16] Concentrating Urine - The Mammalian Kidney. http://www.sumanasinc.com/webcontent/anisamples/majorsbiology/kidney.html. Fecha de consulta: Mayo de 2005
[17] Fox New Media – Nephron animation. http://www.foxnewmedia.de/zms/e16/e233/e62/index_ger.html. Fecha de consulta: Mayo de 2005.
[18] BarcoView project – 3D color volume model. http://www.barco.com/barcoview/downloads/VesselMetrix1.jpg. Fecha de consulta: Mayo de 2005
[19] A.D.A.M. – Body guide - Urinary System. http://www.besthealth.com/besthealth/bodyguide/reftext/html/urin_sys_fin.html. Fecha de consulta: Mayo de 2005
[20] Lexington Community College – The kidney. http://www.uky.edu/LCC/BSN/BIO/BiologyLabs/BSL111/111Lab6/Lab6KidneyModel.html. Fecha de consulta: Mayo de 2005
[21] Rwth-aachen - nephron seminar. http://www.physiology.rwth-aachen.de/lehre/archiv/seminar_niere_pranada/nephron. Fecha de consulta: Mayo de 2005
[22] Galvis Panqueva, Alvaro Hernán. Ingeniería de software educativo. Santa Fe de Bogotá: Uniandes, 1992.
[23] Alvaro h. Galvis-panqueva, D.ed. Micromundos lúdicos interactivos: aspectos críticos en su diseño y desarrollo. Brasilia, Brasil: *RIBIE* 98, 1998.
[24] María Fernanda Aldana Vargas, Martha Luz Arango Muñoz, Diego Ernesto Leal Fonseca, Esperanza López Reyes, Luz Adriana Osorio Gómez, Ana María Salazar Villegas. Metodología para la construcción de ambientes virtuales como apoyo a la educación presencial de la Universidad de los Andes. Santa Fe de Bogotá: Universidad de los Andes.
[25] James & Suzanne Robertson. Volere - Requirements Specification Template - Edition 10.1. http://www.systemsguild.com/GuildSite/Robs/Template.html. London, Aachen & New York, 2004.
[26] T. Alan Keahey. Focus+Context Techniques for Network Visualization. www.visintuit.com 2003
[27] Harald Piringer, Robert Kosara, Helwig Hauser. Interactive Focus+Context Visualization with Linked 2D/3D Scatterplots. VRVis Research Center. http://www.VRVis.at
[28] Kreˇsimir Matkovi´c, Helwig Hauser. Process Visualization with Levels of Detail. VRVis Research Center. http://www.VRVis.at/
[29] L. Mroz – Tiani MedGraph , VR2 – Interacive volume rendering with pc-based virtual reality
A.L. Fuhrmann, B. Özer and H. Hauser – VRVis Research Center
[30] 3D Anatomy for Students. http://sprojects.mmip.mcgill.ca/anatomy3d/. Fecha de consulta: Mayo de 2005
[31] 3D Models. http://www.turbosquid.com/. Fecha de consulta: Mayo de 2005
[32] 3D Models. http://www.3-d-models.com/. Fecha de consulta: Mayo de 2005
[33] Michael Cohen et.al. Wang Tiles for Image and Texture Generation. 2003
[34] Anton Alstes. Wang Tiles for Image and Texture Generation. 2004
[35] Open Source CSG library. http://opencsg.org/. Fecha de consulta: Mayo de 2005
[36] Michael Ross, Wojciech Pawlina and Gordon Kaye, *Histology: A Text and Atlas - 2003* - ISBN: 0683302426
[37] GeoWall project. http://geowall.geo.lsa.umich.edu/. Fecha de consulta: Julio de 2005
[38] Maya. http://www.alias.com/glb/eng/products-services/. Fecha de consulta: Julio de 2005

Umbralización Multinivel Usando el Modelo de Color HSI

César Julio Bustacara Medina
Pontificia Universidad Javeriana, Ingeniería de Sistemas,
Bogotá D.C., Colombia
cbustaca@javeriana.edu.co

y

Enrique González Guerrero
Pontificia Universidad Javeriana, Ingeniería de Sistemas,
Bogotá D.C., Colombia
egonzal@javeriana.edu.co

Abstract

Color usage in images processing is a fundamental factor, because it is a powerful descriptor that simplifies the object identification and extraction in images. Additionally, it is possible to distinguish more objects in color images that gray-level ones. In this paper, one multilevel thresholding technique in color images is proposed, it is based in the following elements: Color images transformation from RGB to HSI color model, to minimize the impact of the illumination conditions; Otsu's method thresholding, based in histograms thresholding technique, is used to find the best optional thresholds; and the k-means clustering technique is used to find the thresholds that will be used in segmentation. The thresholds found with the proposed technique show high confiability with respect to techniques based in RGB color model.

Keywords: Thresholding, Multilevel Thresholding, Otsu's Method, K-Means Clustering, Color Models, Color Images.

Resumen

El uso de color en el procesamiento de imágenes es un factor fundamental, ya que es un potente descriptor que simplifica la identificación y extracción de objetos en una imagen. Además, se puede distinguir una mayor cantidad de objetos en una imagen a color comparado con una imagen a niveles de gris. En este artículo se propone una técnica de umbralización multinivel en imágenes a color, que se basa en los siguientes elementos: transformación de la imagen del espacio de color RGB a HSI, con el fin de minimizar el impacto de las condiciones de iluminación; una técnica de umbralización basada en histogramas usando el método de Otsu para encontrar los mejores candidatos de los umbrales; y la técnica de agrupamiento de K-medias para encontrar las familias de umbrales que serán usados en la etapa de segmentación. Los umbrales encontrados con la técnica propuesta muestran gran confiabilidad con respecto a técnicas basadas en el modelo de color RGB.

Palabras claves: Umbralización, Umbralización multinivel, Método de Otsu, Agrupamiento K-medias, Modelos de color, Imágenes a color.

1 INTRODUCCIÓN

En muchas aplicaciones de procesamiento de imágenes los niveles de gris que representan los objetos son sustancialmente diferentes de los niveles de gris de los píxeles que representan el fondo, es decir, se puede diferenciar el fondo de la imagen de los objetos de interés. La umbralización es una forma que parece simple pero efectiva de separar los objetos del fondo de la imagen. Algunos ejemplos de aplicaciones de umbralización son: el análisis de documentos digitales, donde el objetivo es extraer los caracteres impresos [1][2][3] y contenido gráfico; procesamiento de mapas, donde las líneas, leyendas y caracteres deben ser detectados[4]; inspección y evaluación de la calidad de materiales[5][6], donde los defectos de las partes deben ser identificadas. Otras aplicaciones pueden ser listadas como siguen: representación de conocimiento[7], segmentación de imágenes[8][9][10], análisis de imágenes térmicas y en general todo tipo de aplicación donde se encuentren involucradas imágenes digitales.

La salida de la operación de umbralización es generalmente una imagen binaria, donde un estado indica el fondo y el otro corresponde a los objetos de interés[11]. Dependiendo de la aplicación la asignación de estos valores se puede intercambiar. Varios factores, tales como el ruido, la iluminación, un contraste inadecuado y el tamaño no-medible de los objetos dentro de la escena, complican sustancialmente la operación de umbralización[12].

Sezgin y Sankur[11], desarrollaron una taxonomía para los algoritmos de umbralización, donde no hacen diferencia entre las imágenes de niveles de gris y las de color. Esta clasificación se basa en el tipo de información que usa cada técnica. Las seis categorías que definen son: 1. Basados en la información de la forma del histograma, 2. Agrupamiento del espacio, 3. Basados en la información de la entropía del histograma, 4. Basados en la información de los atributos de la imagen, 5. Basados en la información espacial y 6. Basados en características locales. Sezgin y Sankur también mencionan excelentes referencias de otras taxonomías y algoritmos de umbralización.

Todas las técnicas asociadas a esta taxonomía son bastante funcionales en imágenes con niveles de gris (8-bits equivalentes a 256 colores), pero no funcionan de manera adecuada para las imágenes a color(24-bits, millones de colores), es por ello que aquí se aborda la problemática de umbralización de imágenes a color, ya que permiten discernir un mayor número de elementos dentro de la imagen. Normalmente, un observador típico puede discernir 2 o 3 docenas de niveles de gris, pero si se adiciona color a las imágenes, se pueden distinguir miles de colores y tonos diferentes, por esto el color juega un papel fundamental cuando se usa como descriptor. Adicionalmente, es muy difícil diferenciar niveles de gris adyacentes o cercanos, pero el ojo si puede distinguir entre un gran número de colores. Para distinguir objetos muy próximos en una imagen o para describir o resaltar ciertas características de un objeto es muy útil emplear colores[10].

La propuesta que se explica en este artículo involucra varios métodos de umbralización, los basados en agrupamiento (ya que se utilizará la técnica de k-medias), y los basados en la forma del histograma (técnica de Otsu). Además, de acuerdo a Gevers[13], se utiliza el modelo de color HSI, que permite tener reducción en los efectos que se puedan presentar debido a la variación de la iluminación. En una primera etapa, se convierte la imagen que se encuentra en un modelo de color RGB a HSI, con el fin de obtener invarianza a los efectos de iluminación. En una segunda etapa, se encuentran los histogramas H, S e I unidimensionales (1-D) y los histogramas HS, HI y SI bidimensionales (2-D). Posteriormente, en una tercera etapa se utiliza la técnica de umbralización de Otsu aplicada al histograma H, con el fin de encontrar los mejores candidatos a ser centroides de los grupos de la técnica de k-medias. Finalmente, en una cuarta etapa se hallan los grupos a través del método de k-medias usando el histograma en 2-D HS, lo cual permitirá encontrar los umbrales óptimos de la imagen.

Así, en el numeral 2, se explican con más detalle los modelos de color utilizados, las técnicas de umbralización de Otsu y la técnica de agrupamiento de k-medias. En el numeral 3 se plasma el proceso de umbralización utilizado y en el numeral 4 se muestran algunas de las pruebas realizadas y las conclusiones obtenidas del proceso. Es importante mencionar que parte de las pruebas fueron comparadas con las obtenidas al utilizar el software de umbralización Neurothresholding[14].

2 ANTECEDENTES

2.1 Representación del Color

Existen diferentes modelos para representar la información del color. Dentro de los más utilizados en el procesamiento de imágenes se encuentran: RGB (Red, Green, Blue), HSI(Hue, Saturation, Intensity), HSV (Hue, Saturation, Value), CMY(Cyan, Magenta, Yellow), YIQ(Reflactancia, Infase, Cuadratura) y existen las correspondientes conversiones entre los diferentes modelos enunciados.

2.1.1 Modelo RGB

Este espacio de color esta formado por los colores primarios Rojo, Verde y Azul. Es el adecuado para representar imágenes que serán mostradas en el monitor del computador o que serán impresas en impresoras de papel fotográfico [16][18].

Las imágenes RGB utilizan tres colores para reproducir en pantalla hasta 16,7 millones de colores. RGB es el modo por defecto para representar las imágenes en las herramientas de software para tratamiento de imágenes. Esto es comprensible, ya que si se utiliza un modelo de color diferente para la representación, se debe realizar la conversión a RGB antes de visualizarlo en el monitor del computador [18][19].

El modelo RGB asigna un valor de intensidad a cada píxel, que viene determinado por una tripleta de valores que oscilan entre 0 y 255, que definen un cubo como el que se muestra en la figura 1a. Por ejemplo, un color rojo brillante podría tener un valor R de 246, un valor G de 20 y un valor B de 50. El rojo más brillante que se puede conseguir es el R: 255, G: 0, B: 0. Cuando los valores de los tres componentes son idénticos, se obtiene un matiz de gris. Si el valor de todos los componentes es de 255, el resultado será blanco puro y será negro puro si todos los componentes tienen un valor 0 [17][20].

2.1.2 Modelo HSI

En este modelo, el matiz(H) es un atributo que describe la pureza del color (Amarillo puro, rojo, naranja), mientras que la saturación (S) indica el grado en el que el matiz es diluido con luz blanca. El modelo HSI es muy usado en procesamiento de imágenes debido a dos factores principales: Primero, la componente de iluminación (I), se puede separar de la información de color en la imagen; Segundo, las componentes de matiz y saturación están íntimamente relacionadas con la forma de percepción del color de los seres humanos. Estas características hacen de este modelo una herramienta bastante significativa para desarrollar algoritmos de procesamiento de imágenes basadas en el color. La figura 1b muestra un corte horizontal del doble cono que compone el modelo de color HSI cuando la intensidad es igual a 0.5, en este caso se puede observar que en la periferia aparecen todos los colores puros (S es igual a 1), y cuando se dirige hacia el centro la pureza se va reduciendo[16][18][20][21].

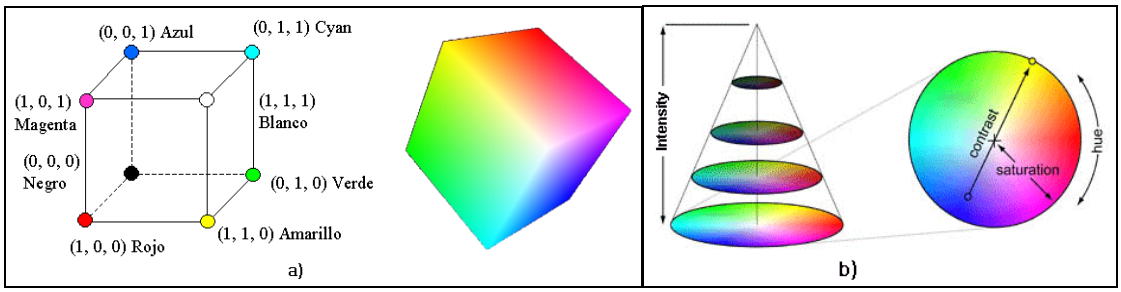


Figura 1 Modelos de color, a) RGB y b) HSI.

2.2 Métodos Automáticos para Encontrar Umbrales

Tanto a nivel global como local se usa frecuentemente el histograma, con el fin de encontrar dos o más regiones diferentes, una para el fondo y las demás para los objetos a segmentar. Se asume que un histograma es la distribución de probabilidad de los diferentes niveles de gris, que esta dada por:

$$p(g) = \frac{n_g}{N}$$

Donde:
n_g, es el número de pixeles que tienen la misma intensidad g, y que
N, es el número total de pixeles de la imagen.

A continuación se presentan cuatro formas diferentes para resolver el problema de encontrar los diferentes umbrales en una imagen, que utilizan los histogramas como elemento fundamental para cumplir con dicha tarea. Pero vale la pena mencionar, que existen modelos mixtos que permiten minimizar el error de clasificación en la umbralización bajo la suposición de que cada grupo es una distribución Gausiana y que cada una de ellas tiene una media μ_i y una desviación estándar $\sigma_{B,}$ independiente del umbral que se seleccione.

2.2.1 Distribución Conocida

Si se conoce que el objeto a segmentar tiene una mayor intensidad que el fondo, y que ocupa una determinada fracción (1/p) de la imagen, se puede colocar el umbral por simple exploración de los niveles de intensidad, tal que, el porcentaje deseado de los pixeles de la imagen sea menor a este valor. Esto es fácilmente extraído del histograma de acumulaciones [13].

$$c(g) = \sum_{0}^{g} p(g)$$

Entonces el umbral T, será tal que:

$$\begin{cases} c(g) = \frac{1}{p}, & \textit{si el objeto es claro} \\ c(g) = 1 - \frac{1}{p}, & \textit{si el objeto es oscuro} \end{cases}$$

2.2.2 Picos y Valles

Una forma efectiva de encontrar un buen conjunto de umbrales es encontrar cada uno de los picos (máximos locales) en el histograma y luego encontrar los valles (mínimos locales) entre ellos, como se muestra en la figura 2.

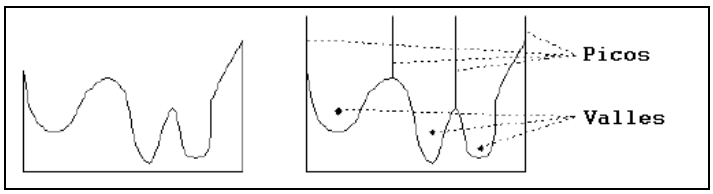


Figura 2. Picos y Valles en un Histograma

Normalmente la imagen posee ruido, lo cual genera una gran cantidad de máximos y mínimos falsos, razón por la cual siempre se sugiere usar un filtro de suavizado antes de encontrar los picos y los valles [13][14].

2.2.3 Agrupamiento (K-Medias)

K-Medias es uno de los algoritmos de aprendizaje no-supervisado más simple que resuelve el problema de agrupamiento. El procedimiento sigue una forma fácil y simple para clasificar un conjunto de datos en una cierta cantidad k de clases o grupos definidas a priori. El problema que surge en ocasiones es el solapamiento de las clases, por ejemplo, cuando un grupo tiene una distribución bastante ancha y el otro bastante estrecha.

La idea principal para calcular las clases o grupos que representaran los umbrales de la imagen es definir k centroides, uno para cada grupo, los cuales deben ser colocados tan lejos como sea posible el uno del otro. Luego se toma cada dato del conjunto y se asocia al centroide más cercano, este proceso se realiza hasta que no queden datos por asignar. A continuación sé recalculan los centroides de cada grupo y se verifica la función objetivo, sino se cumple se vuelve a iterar o de lo contrario se finaliza el proceso y los umbrales finales serán los últimos centroides calculados [22][25][26].

La función objetivo a minimizar, en este caso usando la función de error cuadrática será:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{N} \left\| x_i^{(j)} - c_j \right\|^2$$

Donde $\left\| x_i^{(j)} - c_j \right\|^2$ es la medida de distancia entre un dato $x_i^{(j)}$ y el centroide del grupo c_j.

2.2.4 Agrupamiento (Método de Otsu)

(Otsu, 1979) propuso un algoritmo para seleccionar automáticamente los umbrales a partir del histograma de una imagen a nivel de grises. Para una imagen dada en ***L*** niveles de gris, los correspondientes niveles se representan como el vector [1,2,….L]. El número de píxeles en un determinado nivel ***i***, se denota como $\boldsymbol{n_i}$, y el número total de píxeles de la imagen como N, donde $N = n_1 + n_2 + \ldots + n_i \ldots$. Luego se supone que los píxeles fueron separados en ***k*** clases disjuntas $C_1, C_2,\ldots,C_k$, las cuales denotan los píxeles con niveles $[1,\ldots n_1]$, $[n_1+1,n_2],\ldots,[n_{k-1}+1,\ldots, n_k=L]$, respectivamente. Este método esta basado en un criterio de discriminación, él cual es el radio de varianza entre clases y la varianza total de los niveles de gris [22][23][24][26].

Para cada clase se deben calcular los momentos acumulativos cero y uno, así como la media total para toda la imagen:

$$w_k = \sum_{i=n_{k-1}}^{n_k} p_i \,, \quad \mu_k = \sum_{i=n_{k-1}}^{n_k} \frac{i * p_i}{w_k}, \qquad \mu_T = \sum_{i=1}^{k} \mu_i w_i$$

El momento acumulativo cero representa la probabilidad de que los píxeles pertenezcan a la clase C_i, El primer momento acumulativo representa la media de la clase C_i.

Usando un análisis discriminante, Otsu define la varianza entre clases de la imagen umbralizada como:

$$\sigma_B^2 = \sum_{i=1}^{k} w_i (\mu_i - \mu_T)^2$$

siendo la idea final, encontrar un conjunto optimo de umbrales, $n_i^*, 1 \le i < k$, tales que maximicen σ_B^2 :

$$\{n_1^*, n_2^*, ..., n_{k-1}^*\} = Arg\ Max\{\sigma_B^2(n_1, n_2, ..., n_{k-1})\}, \quad donde \quad 1 \le n_1 \le n_2 \le \cdots \le n_{k-1} < L$$

De acuerdo a la ecuación anterior, para encontrar los umbrales óptimos el rango de búsqueda para maximizar σ_B^2 es $1 \le n_i < L-k+1$, como se muestra en la figura 3. Esta búsqueda implica realizar todas las combinaciones de los intervalos posibles, es decir $(L-k+1)^{k-1}$ posibles combinaciones[24][25].

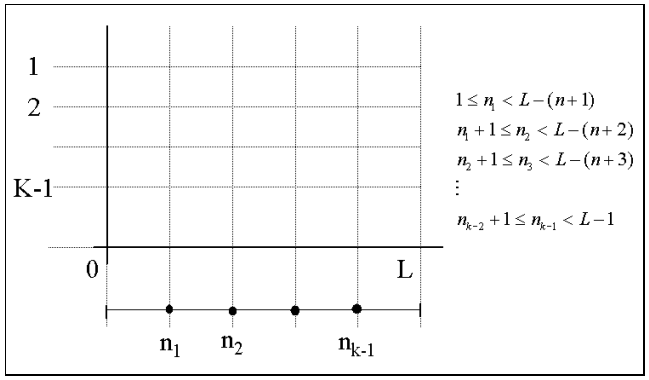


Figura 3 Intervalos de búsqueda para encontrar la solución de Otsu.

2.3 Consideraciones de la Iluminación

Actualmente existen algunos criterios de evaluación de los diferentes modelos de color con el propósito de ser usados en el reconocimiento de objetos a color. Estos criterios son:

- Robustez al cambio del punto de vista de la cámara.
- Robustez al cambio en la geometría del objeto.
- Robustez al cambio en la dirección de la iluminación.
- Robustez al cambio en la intensidad de la iluminación.
- Robustez al cambio en la distribución de potencia espectral (SPD) de la iluminación.

Es importante mencionar que todos estos criterios dependen directamente de las condiciones controladas del espacio de trabajo, pero serán un elemento de decisión en el momento de realizar procesos de procesamiento de imágenes, tal como es la umbralización. Gevers propuso dos modelos adicionales de color $c_1c_2c_3$ y $l_1l_2l_3$[13], que permiten minimizar la influencia de la iluminación en el proceso de reconocimiento de objetos a color.
En la figura 4 se muestran los resultados alcanzados por Gevers, que resumen la influencia de la iluminación tanto blanca como de color y las reflexiones que se producen en el momento de la captura de imágenes.

	Viewing direction	Surface Orientation	Highlights	Illumination direction	Illumination intensity	Illumination Color	Inter-Reflection
I	-	-	-	-	-	-	-
RGB	-	-	-	-	-	-	-
rgb	+	+	-	+	+	-	-
S	+	+	-	+	+	-	-
$c_1c_2c_3$	+	+	-	+	+	-	-
H	+	+	+	+	+	-	-
$l_1l_2l_3$	+	+	+	+	+	-	-
$m_1m_2m_3$	+	+	-	+	+	+	+

Figura 4 Resumen de varios modelos de color y su invarianza a las condiciones. + denota invarianza y - denota sensibilidad del modelo de color a las condiciones.

En la figura 4 se puede observar que el modelo RGB es bastante sensible a la iluminación, mientras que las componentes H y S del modelo HSI son poco sensibles, razón por la cual serán usados para definir los umbrales más adecuados para las imágenes. Es claro que el modelo *rgb* que corresponde al modelo normalizado de RGB también es un buen candidato para este mismo proceso[13].

3 ALGORITMO PROPUESTO PARA UMBRALIZACIÓN MULTINIVEL

El proceso de umbralización de imágenes de objetos en entornos parcialmente controlados muchas veces es compleja por la falta de conocimiento a priori del número de objetos a detectar, por la influencia de elementos no deseados como sombras, brillos, complejidad de colores, texturas, tamaño de los objetos, posibilidad de solapamiento de los objetos, variaciones en el fondo, y si la captura se hace a través de una cámara, el ruido incorporado. Todos estos elementos influyen de manera directa en el número de umbrales detectados, ya que se puede presentar una sub-umbralización, es decir, se detectan menos regiones de interés, lo cual influirá en la etapa siguiente del proceso de visón, que corresponde a la segmentación. Aunque también se puede presentar el caso opuesto, que implicaría la detección de más regiones de interés de las existentes en la imagen, conllevando a un proceso de sobre-umbralización.

El algoritmo propuesto consta de varios pasos, como se muestra en la figura 5, los cuales se ejecutan de forma secuencial hasta encontrar los umbrales multinivel que permitirán realizar la segmentación. Cada uno de los pasos se explica en detalle más adelante. En primera instancia se aplica el filtrado de la imagen, usando el filtro no-lineal de la mediana que permite suavizar la imagen y así tener una representación más uniforme de las regiones a umbralizar, luego, aparece la conversión del modelo de color RGB a HSI, posteriormente, se encuentran los histogramas en 1-D y 2D (H, S, I, HS, HI y SI), a continuación se aplica la umbralización multinivel usando la técnica de Otsu al histograma unidimensional H para encontrar los mejores candidatos de centroides, finalmente, usando estos centroides se aplica la técnica de agrupamiento de k-medias. Existe un paso intermedio, que depende del fondo de la imagen y consiste en realizar una umbralización unimodal, con el fin de eliminar los elementos de baja o alta saturación en HSI, esto permite tener una mayor confiabilidad de los umbrales encontrados, es decir, si el fondo es negro, se eliminan las contribuciones de bajas saturaciones y si el fondo es blanco se eliminan los elementos con las altas saturaciones. Pese a que este paso hace del algoritmo propuesto una técnica semiautomática, aumenta el nivel de confiabilidad de la técnica.

Figura 5 Algoritmo propuesto.

3.1 Conversión del modelo RGB a HSI

El método propuesto trata de minimizar los problemas de sub-umbralización y sobre-umbralización, realizando el proceso de multi-umbralización en el histograma bidimensional HS. Para ello en primera instancia se convierte la imagen de color de RGB a su modelo normalizado *rgb*, es decir, se busca minimizar el impacto de la iluminación y de las sobras que se puedan presentar en la imagen, para ello se utilizan las ecuaciones siguientes[13]:

$$r = \frac{R}{R+G+B}$$

$$g = \frac{G}{R+G+B} \qquad donde, \quad 0 \le R,G,B \le 255 \quad y \quad 0 \le r,g,b \le 1$$

$$b = \frac{B}{R+G+B}$$

Posteriormente, se convierte el modelo RGB normalizado(rgb) a HSI, utilizando las siguientes ecuaciones:

$$H = tan^{-1}(\frac{\sqrt{3(g-b)}}{(r-g)+(r-b)})$$

$$I = \frac{r+g+b}{3}$$

$$S = 1 - \frac{min(r,g,b)}{I}$$

Además, el modelo HSI minimiza el impacto de los objetos de color negro y blanco, ya que aparecen en los extremos del cono de color HSI. La componente H (matiz) permite encontrar los posibles colores que conforman la imagen, y la componente S, el nivel de saturación (dilusión del color en luz blanca). En teoría, con estas dos componentes se pueden encontrar los umbrales, pero en la práctica funciona diferente, porque el valor de iluminación hace que los umbrales sean más o menos precisos. Por tal motivo, no se debe dejar de lado en el momento de hacer la conversión de HSI a RGB para representar los umbrales de la imagen.

3.2 Cálculo de los histogramas en 1-D y 2-D

La evaluación de los histogramas en una dimensión, se puede reducir a encontrar el histograma del matiz (Hue), ya que a partir de este se evalúan los centroides iniciales, pero este no es suficiente, ya que es necesario calcular el histograma de la saturación para identificar el umbral unimodal que permitirá reducir la influencia de la iluminación. Además, el histograma de la Iluminación sé evalúa para ratificar la decisión del umbral de saturación, es decir, si los niveles de iluminación son muy bajos, implica que la saturación es alta, esta conclusión se obtiene de la inspección de la forma del modelo de color HSI, ya que si la iluminación se acerca a 1 o a 0 conlleva a deducir que la saturación es alta, debido a que el radio del circulo de color asociado se reduce.

Los histogramas en dos dimensiones se calculan para realizar el proceso de agrupamiento, en el algoritmo propuesto se halla el histograma HS, con el fin de encontrar los umbrales del matiz y la saturación, y luego con los valores encontrados se pasa a evaluar el histograma HI y así, encontrar los valores de la saturación e iluminación apropiados para regresar los umbrales al modelo de color RGB. Es necesario el paso final para no tener que realizar el proceso de conversión en el momento que se realice la segmentación.

3.3 Umbralización Multinivel de Otsu y Agrupamiento de K-Medias

El método propuesto por Otsu, permite encontrar con precisión los umbrales de forma unimodal y multimodal. Aunque existen estudios sobre la conversión del método multinivel de Otsu de 1-D a 2-D[15][27], los resultados no son óptimos. Es por ello que únicamente son usados en el algoritmo propuesto para hallar los candidatos iniciales de los centroides. Se realizaron varias pruebas para justificar la utilización del espacio HSI, los cuales indicaron que era la mejor alternativa, ya que el histograma unidimensional del matiz arroja mayor información que los canales Rojo, Verde y Azul del espacio RGB. La figura 6 muestra una imagen inicial que posee colores bien definidos y los correspondientes histogramas en 1-D, la figura 7 muestra la misma imagen y los histogramas en 2-D, (fue necesario adicionar círculos alrededor de los puntos de acumulación para su observación), concluyéndose que los histogramas en HSI poseen mayor información del color de la imagen.

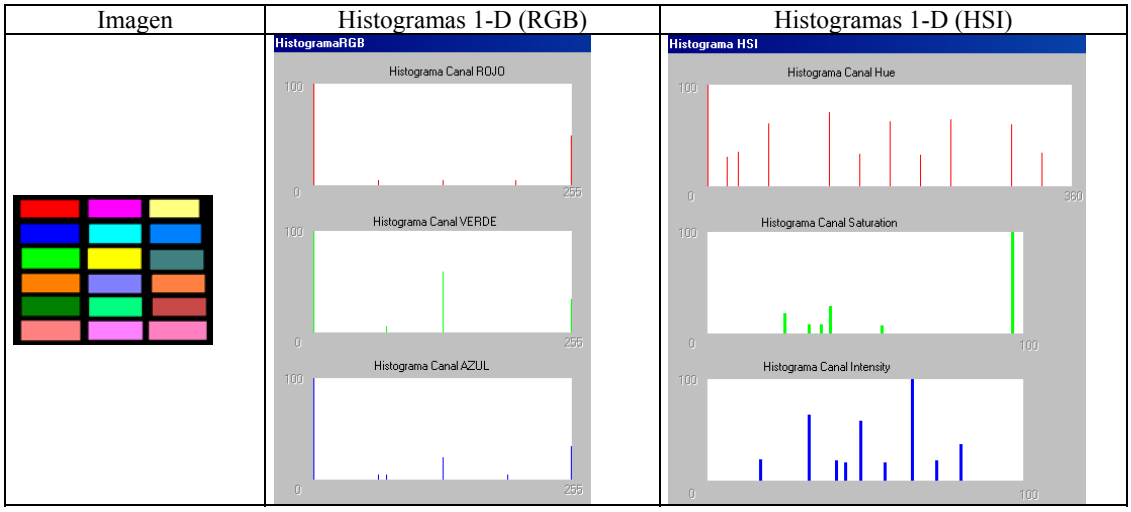


Figura 6 Histogramas en 1-D para RGB y HSI.

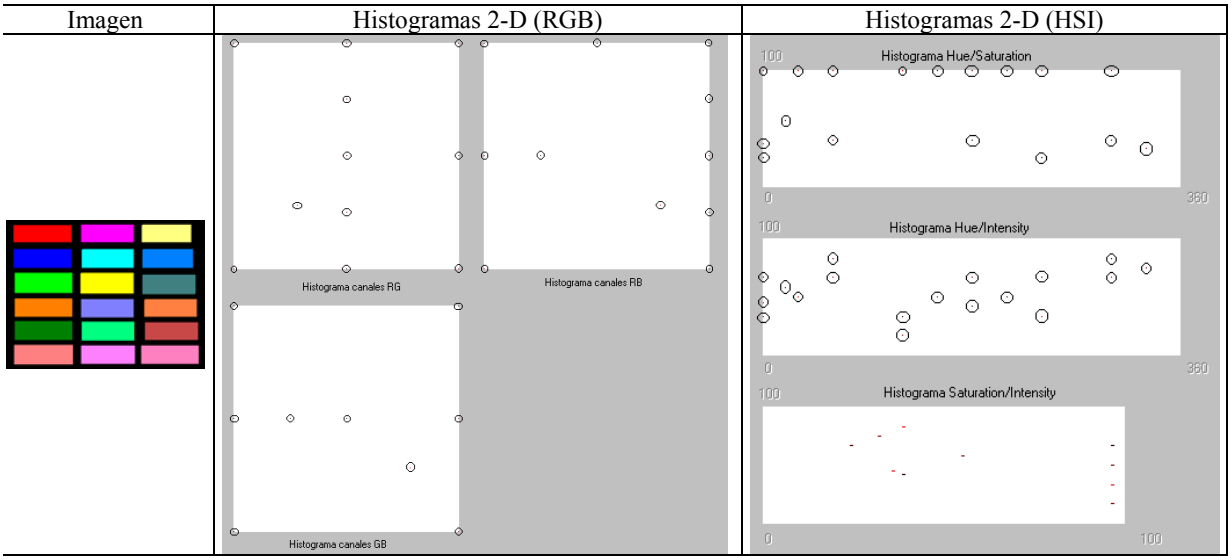


Figura 7 Histogramas en 2-D para RGB y HSI.

La figura 8 y 9 muestran respectivamente, la consecuencia de aplicar un filtro mediana unitario a la imagen para encontrar de una manera más precisa los umbrales multimodales en 1-D y en 2-D.

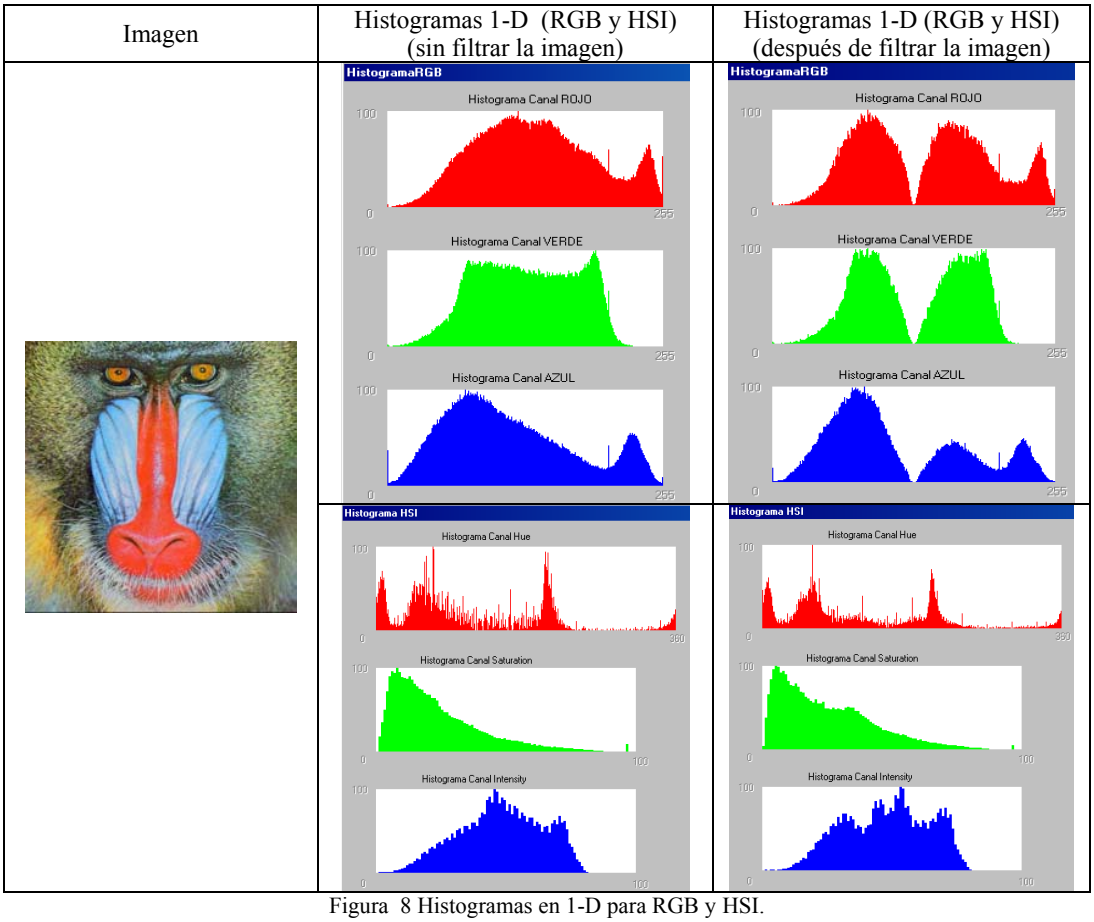


Figura 8 Histogramas en 1-D para RGB y HSI.

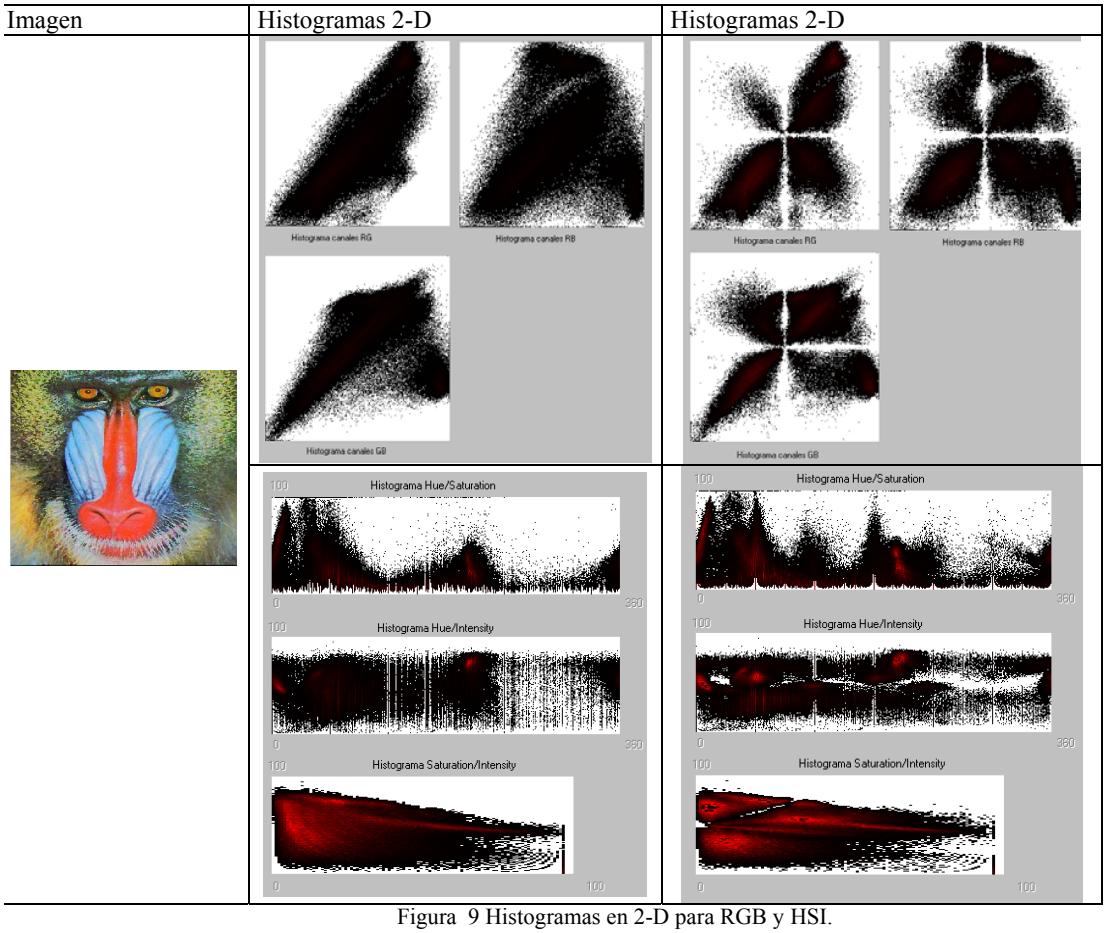


Figura 9 Histogramas en 2-D para RGB y HSI.

Además, se decidió aplicar un filtro pasa bajas, con el fin de suavizar el histograma y así reducir la aparición de falsos picos y por ende de falsos umbrales. El filtro utilizado fue un filtro promedio de tamaños 3 y 5, dando mejores resultados el filtro de tamaño 3.

Finalmente, se utiliza uno de los algoritmos más tradicionales para realizar agrupamiento de datos en *k* clases: el algoritmo de k-medias. El cual permite codificar cada uno de los datos (pixeles) buscando minimizar la suma de los cuadrados de la distancia de cada pixel con respecto al centroide.

4 PRUEBAS Y RESULTADOS

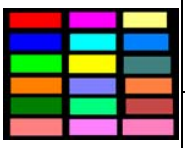
Se realizaron varias pruebas, tanto a nivel controlado como no controlado. El grupo de pruebas se dividió en tres frentes de acuerdo al tipo de imágenes utilizadas, primero se tomaron imágenes construidas a través de un editor gráfico, que permite controlar valores de las componentes R, G y B de la imagen. La finalidad esencial era definir un patrón en común para poder evaluar las dos técnicas de umbralización. Un ejemplo de este tipo de imágenes es el que aparece en la figura 6. En segunda instancia, se usaron imágenes tradicionales en tratamiento de imágenes con la finalidad de comparar los resultados de forma cualitativa, y finalmente, se tomó un tercer grupo de imágenes directamente digitalizadas a través de una cámara Javelin y una tarjeta de adquisición Matrox Meteor II, con la finalidad de evaluar en un entorno real el comportamiento de los algoritmos. En este punto, fue necesario para utilizar el software de NeuroThresholding, salvar las imágenes a disco previamente.

La figura 10 muestra un ejemplo de pruebas con imágenes controladas (construidas), donde se puede observar que tanto el algoritmo propuesto en HSI, como el utilizado en RGB, generan perdida en los umbrales que se desean encontrar, pese a que se escogió un conjunto de colores bien separados. En la figura 10 se muestra la lista de los valores de los

umbrales ideales, con los obtenidos, y la reorganización de los obtenidos para identificar las redundancias de umbrales. Se puede observar que el modelo RGB funciona mejor que el HSI, razón por la cual fue necesaria la incorporación de la umbralización unimodal descrita en el numeral 3.

Modelo: RGB

Imagen:



Valores ideales			Obtenidos				Valores reordenados		
Blue	Green	Red	Blue	Green	Red				
0	0	0	0	0	0	ok	0	0	0
0	0	255	0	0	0	invalida			
0	128	0	0	55	0	invalida	0	128	0
0	128	255	0	61	255	ok	0	61	255
0	255	0	0	128	0	ok	0	255	0
0	255	255	0	255	0	ok	0	255	255
64	128	255	0	255	255	ok	68	101	228
72	72	200	68	101	228	ok	128	181	255
128	128	64	128	181	255	ok			
128	128	255	128	255	0	ok			
128	255	0	152	205	22	invalido	128	255	0
128	255	255	182	182	37	invalido			
192	128	255	192	128	255	ok	192	128	255
255	0	0	253	72	76	invalido	255	0	0
255	128	0	255	0	0	ok	255	128	0
255	128	128	255	68	255	ok	255	128	128
255	128	255	255	128	0	ok	255	68	255
255	255	0	255	128	128	ok			

Modelo: HSI

Valores ideales			Valores Obtenidos				Valores Reordenados		
Blue	Green	Red	Blue	Green	Red		Blue	Green	Red
0	0	0	0	0	0	ok	0	0	0
255	0	0	0	0	0	invalido	252	59	0
0	128	0	252	59	0	ok	0	202	0
255	128	0	0	202	0	ok			
0	255	0	188	255	0	ok			
128	255	0	129	128	64	ok	188	255	0
255	255	0	44	45	70	invalido			
128	128	64	92	238	126	invalido	129	128	64
255	128	128	254	128	128	ok	254	128	128
72	72	200	0	177	145	invalido	72	72	201
0	0	255	233	121	196	ok			
0	128	255	233	121	196	invalido	0	137	255
64	128	255	72	72	201	ok	97	125	255
128	128	255	255	62	208	ok			
192	128	255	255	0	253	invalido			
255	128	255	97	125	255	ok	233	121	196
0	255	255	0	137	255	ok	0	177	145
128	255	255	128	255	255	ok	128	255	255

Figura 10 Ejemplo de imágenes controladas y sus correspondientes umbrales.

Finalmente, en este articulo se muestra una de las pruebas de umbralización con imágenes capturadas en la figura 11. A primera vista muestra la ventaja de usar el algoritmo propuesto frente al de NeuroThresholding, ya que del proceso de umbralización se identifican todas las regiones de interés de la escena. Claro esta que con esta imagen y sus condiciones de captura, tiene un problema para diferenciar los dos tonos de rojo debido a su proximidad, esto se ve en la tercera imagen de izquierda a derecha de la figura 11, la cual muestra el proceso paso a paso de los umbrales encontrados y su consecuencia en la imagen capturada. Este problema se soluciona modificando la iluminación de la escena y los parámetros de captura de la cámara. En la figura 12, se muestra la tabla de valores deseados, de acuerdo a condiciones ideales de captura, los obtenidos en RGB y los obtenidos en HSI, así como el error en cada uno de los canales respecto al valor ideal. Se observa que en promedio el error es menor al usar el método propuesto en HSI que al utilizar el de RGB.

5 CONCLUSIONES

Se ha desarrollado con éxito una aplicación que aplica el algoritmo propuesto para la umbralización usando el espacio de color HSI. Los resultados obtenidos con el algoritmo son satisfactorios para poder realizar un proceso de segmentación basado en umbrales. No obstante queda abierta la posibilidad de mejorar el desempeño del algoritmo, ya que este método adiciona dos pasos de preprocesamiento a nivel de imagen y a nivel de histograma, así como la necesidad de realizar una umbralización unimodal usando el histograma S y la técnica de Otsu.

Un punto importante que surgió del proceso, fue el encontrar que la iluminación juega un papel importante en la conversión desde los umbrales encontrados en HSI a RGB, ya que la mayoría de autores omiten esta componente, lo cual genera una baja precisión en los umbrales obtenidos, esta sensibilidad se soluciona usando la umbralización unimodal ya mencionada. El solapamiento de umbrales se solucionó de manera indirecta al usar el modelo de color HSI, y al ubicar los grupos o clases de umbrales en el histograma HS. Algo que surgió de manera inesperada, fue la necesidad de agrupamiento en el histograma HI y SI, con la finalidad de solucionar el problema del desconocimiento de la intensidad.

Imagen Capturada	Umbralización RGB	Umbralización Propuesta

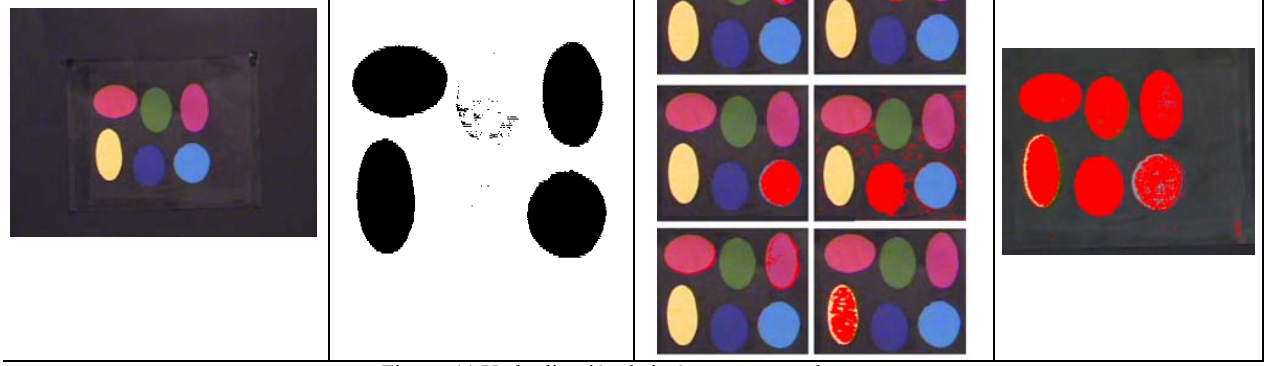


Figura 11 Umbralización de imágenes capturadas.

Deseado			NeuroThesholding			Algoritmo Propuesto			Error			Error RGB	Error			Error Propuesto
Blue	*Green*	*Red*	*Blue*	*Green*	*Red*	*Blue*	*Green*	*Red*	*Blue*	*Green*	*Red*		*Blue*	*Green*	*Red*	
60	110	90	48	52	56	70	142	121	12	58	34	**104**	10	32	31	**73**
100	75	200	54	48	56	86	182	198	46	27	144	**217**	14	107	2	**123**
120	50	190	59	45	51	100	71	176	61	5	139	**205**	20	21	14	**55**
130	220	250	92	162	168	122	187	206	38	58	82	**178**	8	33	44	**85**
140	60	70	113	65	190	131	62	84	27	5	120	**152**	9	2	14	**25**
210	130	90	186	91	74	190	110	76	24	39	16	**79**	20	20	14	**54**
Error Total									208	192	535	**935**	81	215	119	**415**

Deseado			NeuroThesholding			Algoritmo Propuesto			Error			Error RGB	Error			Error Propuesto
Blue	*Green*	*Red*	*Blue*	*Green*	*Red*	*Blue*	*Green*	*Red*	*Blue*	*Green*	*Red*		*Blue*	*Green*	*Red*	
60	110	90	60	47	63	70	75	140	0	63	27	**90**	10	35	50	**95**
100	75	200	97	106	96	76	138	166	3	31	104	**138**	24	63	35	**121.5**
120	50	190	128	84	176	98	137	59	8	34	14	**56**	22	87	131	**240**
130	220	250	139	163	251	120	190	210	9	57	1	**67**	10.5	30	40	**80.5**
140	60	70	182	113	92	143	66	85	42	53	22	**117**	2.5	6	15	**23.5**
210	130	90	239	235	253	199	157	83	29	105	163	**297**	11.5	27	7	**45.5**
Error Total									91	343	331	**765**	80.5	248	278	**606**

Figura 12 Valores de error entre los umbrales deseados y los obtenidos usando RGB y HSI.

6 REFERENCIAS

[1] Kamel M., Zhao A., Extraction of binary characters/graphics images from grayscale document images, *Graph. Models Image Process*. 55(3), 203-217 (1993).

[2] Abak T., Baris U., Sanku B., The performance of thresholding algorithms for optical character recognition, *International Journal on Document Analysis and Recognition*. ICDAR'97, pp 697-700 (1997).

[3] Thouin P., Chang C., A method for restoration of low-resolution text images, *International Journal on Document Analysis and Recognition*, Vol. 2 No. 4, (June 2000), pp 200-210.

[4] Trier O., Jain A., Goal-directed evaluation of binarization methods*, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-17*, 1191-1201 (1995).

[5] Sezgin M., Tasaltin R., A new Dichotomization technique to multilevel thresholding devoted to inspection applications, *Pattern Recognition Letter 21*, 151-161 (2000).

[6] Sezgin M., Sankur B., Comparison of thresholding methods for non-destructive testing applications, *IEEE ICIP'2001, International Conference Image Process,* (2001), pp. 764-767.

[7] Bock P., Klinnert R., Kober R., Rovner R., Schmidt H., Gray-scale ALIAS, *IEEE Transactions Knowledge Data Eng. 4*, 109-122 (1992).

[8] Kohler R., Asegmentation system based on thresholding, *Graph. Models Image Process. 57*, 319-338 (1981).

[9] Perez A., Pavlidis T., An iterative thresholding algorithm for image segmentation, *IEEE Transaction Pattern Anal. Mach. Intell. PAMI-9*, 742-752(1987).

[10] Gil, P. and Torres F. and Ortiz F.G, Detección de objetos por segmentación multinivel combinada de espacios de color. *XXV Jornadas de Automática, Ciudad Real*. (Septiembre 2004).

[11] Sezgin M., Sankur B., Survey over image thresholding techniques and quantitative performance evaluation, *Journal of Electronic Imaging 13(1)*, 146-165 (January 2004).

[12] Wang H., A survey of thresholding techniques, (March 1997).

[13] Gevers T., Smeulders A., Color based object recognition, *Elsevier*, (Noviembre 1999).

[14] Página del Profesor Nikos Papamarkos, Free Image Processing Software, http://ipml.ee.duth.gr/~papamark/free_software2.htm, Ultima fecha de consulta: Mayo 26 de 2005.

[15] Liao P., Chen T., Chung P., A fast algorithm for multilevel thresholding, Journal of Information Science and Engineering 17, (2001), pp. 713-727.

[16] Hubel D., Eye brain and vision, New York: Scientific American Library, 1988.

[17] Tony F. Chana, Sung Ha Kanga and Jianhong Shenb, Total Variation Denoising and Enhancement of Color Images Based on the CB and HSV Color Models, Journal of Visual Communication and Image Representation Volume 12, Issue 4 , December 2001, Pages 422-435.

[18] Shashi D. Buluswara and Bruce A. Draperb , Color Models for Outdoor Machine Vision, Computer Vision and Image Understanding, Volume 85, Issue 2 , February 2002, Pages 71-99.

[19] Helman Stern and Boris Efros, Adaptive color space switching for tracking under varying illumination, Image and Vision Computing, Volume 23, Issue 3 , 1 March 2005, Pages 353-364.

[20] Thomas Wachtlera, , , Ulrike Dohrmannb and Rainer Hertela, Modeling color percepts of dichromats, Vision Research,Volume 44, Issue 24 , November 2004, Pages 2843-2855.

[21] J. B. Mena, , and J. A. Malpica, Color image segmentation based on three levels of texture statistical evaluation, Applied Mathematics and Computation, Volume 161, Issue 1 , 4 February 2005, Pages 1-17.

[22] Jain A., Murty M., Flynn P., Data Clustering: A Review, Based on the book: Image Segmentation Using Clustering, IEEE Computer Society Press, 1996.

[23] Cheriet M., Said J., Suen C., A recursive Thresholding Technique for image Segmentation, IEEE Transactions On Image Processing, Vol 7 No. 6 June 1998.

[24] Du Y., Chang C., Thouin P., Unsupervised approach to color video thresholding, Society of Photo-Optical Instrumentation Engineers, February 2004.

[25] Kanungo T., Netanyahu N., Wu A., An efficiente k-Means Clustering Algorithm: Analysis and Implementation, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 24, No 7, July 2002.

[26] Cheung Y., K*-Means: A new generalized k-means clustering algorithm, Pattern Recognition Letters 24 (2003).

[27] Jianzhuang L., Wenging L., Yupeng T., Automatic Thresholding of gray-level Pictures Using Two-Dimensional Otsu Method, International Conference on Circuits and Systems, June 1991, China.

Sistemas Multi-Agente ubícuos para la recuperación de información adaptada

Angela Carrillo Ramos, Jérôme Gensel, Marlène Villanova-Oliver, Hervé Martin
LSR-IMAG Laboratory, SIGMA Team. B.P. 72
38402 Saint Martin d'Hères Cedex, France
Fax: +33 (0) 476827287
{carrillo, gensel, villanov, martin}@imag.fr

Abstract

Nowadays, through wireless networks, access can widely be made to *Web-based Information Systems (WIS)* by nomadic users. However, *Mobile Devices (MDs)* are still characterized by intrinsic reduced capacities (e.g., size of screen, memory, data storage…) which must be taken into account by *WIS* designers in order to render this access handy. *PUMAS* (*Peer Ubiquitous Multi-Agent System*) is a framework based on four *Multi-Agent Systems* whose main objective is to provide nomadic users with information adapted to different criteria (e.g., preferences, location). In *PUMAS*, when users search for information through their *MD*, their query is modified by different intelligent agents which add to it some information related to the characteristics of both the users and their *MDs*. Once the augmented query comes to the *Router Agent*, this agent performs a *Query Routing* process which consists in searching the information sources which can answer the query taking into account the users preferences, the features of their *MDs*, their location, etc. This search is based on the knowledge managed and exchanged by the *PUMAS* agents (especially, those belonging to the *Information* and the *Adaptation MAS*). In this paper, we describe the different kinds of knowledge managed by the *PUMAS* agents and the way *PUMAS* performs the *Query Routing* process in three steps: analysis of the query, selection of the information sources and redirection of the query.

Keywords: Multi-Agent Systems, Ubiquitous Computing, Adaptation, Query Routing.

Resumen

Hoy en día, un usuario nómada puede acceder *Sistemas de Información Web (SIW)* a través de redes inhalámbricas. Sin embargo, los dispositivos de acceso que se utilizan (*Dispositivos Móviles, DMs)* se caracterizan por sus intrínsecas capacidades reducidas (por ejemplo, tamaño de la pantalla, memoria, almacenamiento). Los diseñadores de *SIW* deben tener en cuenta estas capacidades con el fin de facilitar al usuario dicho acceso. *PUMAS (**P**eer **U**biquitous **M**ulti-**A**gent **S**ystem)* es un framework basado en cuatro *Sistemas Multi-Agentes (SMAs)* cuyo principal objetivo es proveer a usuarios nómadas con información adaptada según diferentes criterios: sus preferencias, su localización, etc. En *PUMAS*, cuando un usuario nómada busca información a través de su *DM*, su consulta es modificada por diferentes agentes inteligentes que le adicionan información relacionada con las características tanto del usuario como de su *DM*. Una vez la consulta "*aumentada"* llega al *Router Agent* (que pertenece al *SMA de Información* de *PUMAS*), este agente ejecuta el proceso de *enrutamiento de consultas* (*Query Routing*), que consiste en la búsqueda de las fuentes de información más adecuadas que puedan responder dichas consultas teniendo en cuenta aspectos como las preferencias del usuario, las características de su *DM*, su localización, etc. Esta búsqueda se basa en el conocimiento manejado e intercambiado por los agentes de *PUMAS* (especialmente, aquellos que pertenecen a los *SMA* de *Información* y de *Adaptación*). En este artículo describimos el conocimiento manejado por los agentes de *PUMAS* y la forma en que llevan a cabo el proceso de *Query Routing* en tres etapas: el análisis de la consulta, la selección de fuentes de información y la redirección de la consulta.

Palabras claves: Sistemas Multi-Agente, Computación Ubícua, Adaptación, Enrutamiento de Consultas.

1. INTRODUCTION

Las aplicaciones que se ejecutan sobre *Dispositivos Móviles (DMs)* son diseñadas para permitir a los usuarios la consulta de datos en cualquier momento desde cualquier lugar. Esta es la idea de base de la *Computación Ubícua (Ubiquitous Computing)* definida por la *W3C* [12] como un paradigma emergente de computación personal que se caracteriza por el tipo de dispositivos de acceso utilizados (dispositivos de cómputo pequeños, inhalámbricos, que se manipulan fácilmente con una o dos manos). Estos dispositivos requieren arquitecturas de red capaces de soportar su configuración automática y ad-hoc, que tomen en cuenta las características del ambiente en el que se desarrolla la *computación ubícua*

(*ubiquitous computing environment)* tales como heterogeneidad, mobilidad, autonomía, alta distribución, etc. Pirker *et al* [9] definen dicho ambiente como una red dinámica de sistemas y de dispositivos embebidos que pueden interactuar con el usuario con el objetivo de satisfacer sus requerimientos y proveerlos con una variedad de servicios colaborativos de información y de comunicación.

Koch *et al* [5] muestran cómo la *computación ubícua* se fortalece cuando la aplicación tiene la inteligencia para procesar información contextual (es decir, localización, tiempo de conexión...) acerca del usuario y de su entorno con el fin de brindarle la información adecuada en el lugar y momento adecuados. Adicionalmente, se busca que la aplicación ofrezca la posibilidad de tomar cierta iniciativa en representación del usuario (características *proactivas* y *adaptativas*). Dichos autores recomiendan el uso de la *tecnología agente* en *aplicaciones ubícuas* ya que un sistema de cómputo móvil debe, en primer lugar, ser *proactivo*, es decir, que razone sobre las actividades del usuario y que analice cómo estas pueden ser ejecutadas. En segundo lugar, el sistema debe ser *adaptativo* debido a que el ambiente de *computación ubícuo* es altamente dinámico (los usuarios pueden desplazarse de un lugar a otro y sus actividades pueden cambiar basadas en sus características contextuales). Con el fin de proveer al usuario nómada sólo con la *información más relevante* ("*la información adecuada, en el lugar y momento adecuados*"), una aplicación que se ejecuta en *DMs* debe contar con mecanismos para propagar las consultas (*queries*) de los usuarios hacia las fuentes de información más apropiadas (almacenadas en uno o más dispositivos) que respondan estas consultas teniendo en cuenta las preferencias del usuario, las características de su *DMs*, su localización, etc. Este es el principal propósito del proceso de *enrutamiento de consultas* (*Query Routing*). Xu *et al* [10] lo definen como el problema general de evaluar la consulta usando las fuentes de información más relevantes y de integrar los resultados provenientes de dichas fuentes. Para optimizar este proceso, los trabajos presentados en [1] y [8] proponen usar ciertas métricas relacionadas con la confianza en las fuentes de información, su capacidad para satisfacer las necesidades de información de los usuarios y los tiempos de respuesta.

En [3], presentamos *PUMAS*, un *framework* cuyo objetivo principal es el de recuperar de diferentes fuentes, la información que se ajuste a las necesidades del usuario (de acuerdo a su perfil y a las características técnicas de su *DM*). La arquitectura de *PUMAS* se compone de cuatro *Sistemas Multi-Agente (SMA)*, cada uno conformado por uno o más agentes ubícuos (*ubiquitous agents*) que cooperan con el fin de ofrecer diferentes servicios: conexión/desconexión de los *DMs*, búsqueda de información en diferentes fuentes, etc. En *PUMAS* se utilizan archivos *XML* para representar e intercambiar información entre agentes (datos, roles de agentes y reglas). El presente artículo tiene como puntos centrales tanto la representación del conocimiento manejado por los agentes de *PUMAS* con el fin de adaptar la información, como el proceso de *Query Routing.* Dicho proceso es ejecutado por el *Router Agent* (cuyo rol se define aquí) y consiste en redirigir las consultas formuladas por el usuario hacia diferentes *Sistemas de Información Web (SIW*s). Mostramos aquí cómo se usan en dicho proceso las *Bases de Conocimiento* de los agentes de *PUMAS.*

La estructura de este artículo es la siguiente: Introducimos la arquitectura de *PUMAS* en la sección 2. Luego en la sección 3 mostramos algunos escenarios en los que se describe la conexión de un usuario a *PUMAS*, así como el envío de una consulta y la posterior recepción de sus resultados. En la sección 4 describimos las *piezas de conocimiento* (que en este trabajo llamamos *"hechos"*) utilizadas por los agentes de *PUMAS* para adaptar la información a las características del usuario y de su *DM*; mostramos particularmente los hechos utilizados por los agentes de los *SMAs* de *Información* y de *Adaptación*. En la sección 5 mostramos un ejemplo del uso de nuestro *framework* en un *SIW* de un hospital. En la sección 6, describimos el proceso de *Query Routing* ejecutado por el *Router Agent* antes de concluir.

2. UN FRAMEWORK LLAMADO PUMAS

La arquitectura de *PUMAS* está compuesta de cuatro *Sistemas Multi-Agente* (ver Figura 1):

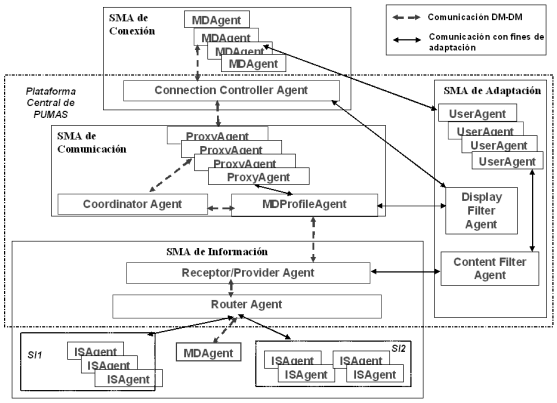


Figura 1. Arquitectura de PUMAS

El *SMA de Conexión* provee los mecanismos que facilitan la conexión de diferentes tipos de *DMs* al sistema (plataforma central de *PUMAS*).

El *SMA de Comunicación* garantiza una comunicación transparente entre los *DMs* y el sistema. También aplica el *Filtro de Despliegue* que consiste en mostrar al usuario a través de su *DM*, la información de acuerdo a las restricciones técnicas de su *DM*. Para hacer esto, este *SMA* cuenta con la ayuda de los agentes del *SMA de Adaptación*.
El *SMA de Información* recibe las consultas (*queries*) del usuario y las redirecciona hacia el *Sistema de Información* (*SI*) más "*adecuado*" (e.g.,el más cercano, o el más consultado), aplica el *Filtro de Contenido* con la ayuda del *SMA de Adaptación*, y de acuerdo al perfil del usuario en el sistema, retorna los resultados "*filtrados*" al *SMA* de *Comunicación*.
Los agentes del *SMA de Adaptación* se comunican con los agentes de los otros tres *SMAs* con el fin de intercambiar información acerca del usuario (explícitamente extraída de los archivos *XML* o inferida de las reglas que administran el sistema), acerca de la conexión y comunicación, las características del *DM*, etc. Los servicios y tareas desempeñados por el *SMA* de *Adaptación* consisten esencialmente en el manejo de archivos *XML* específicos que contienen información sobre el usuario y su *DM*. Los agentes del *SMA de Adaptación* manejan conocimiento (almacenado en *Bases de Conocimiento - BC*) que permite filtrar la información a los usuarios. Existe también conocimiento que es inferido a partir del análisis de la historia del usuario en el sistema, incluyendo sus últimas conexiones, consultas, preferencias, etc.
Una descripción más detallada de los componentes de *PUMAS* se encuentra en [3]. En el presente artículo se introduce y se detalla tanto el manejo como el intercambio de información de los agentes de *PUMAS* (ver siguientes subsecciones).

2.1 El SMA de Conexión

Este *SMA* incluye uno o más *Mobile Device Agents* y un *Connection Controller Agent*.

```
<?xml versión="1.0"?> <rdf:RDF <owl:Ontology rdf:about=""/>
 <owl:Class rdf:ID="Perfil-MIDP-PJava">
  <rdfs:subClassOf> <owl:Class rdf:ID="SoftwareRequirements"/> </rdfs:subClassOf> </owl:Class>
 <owl:Class rdf:ID="kVM">
  <rdfs:subClassOf> <owl:Class rdf:about="#SoftwareRequirements"/> </rdfs:subClassOf></owl:Class>
 <owl:Class rdf:ID="Resolution">
  <rdfs:subClassOf> <owl:Class rdf:ID="Screen"/> </rdfs:subClassOf> </owl:Class>
 <owl:Class rdf:about="#Screen">
  <rdfs:subClassOf> <owl:Class rdf:about="#HardwareRequirements"/> </rdfs:subClassOf>  </owl:Class>
 <owl:Class rdf:ID="Sound">
  <rdfs:subClassOf> <owl:Class rdf:ID="SupportedHyperMediaTypeFiles"/> </rdfs:subClassOf>  </owl:Class>
 <owl:Class rdf:ID="DisconnectionType">
  <rdfs:subClassOf> <owl:Class rdf:ID="DisconnectionStatus"/> </rdfs:subClassOf> </owl:Class>
 <owl:Class rdf:ID="InactivationBeforeDisconnection">
  <rdfs:subClassOf> <owl:Class rdf:about="#DisconnectionStatus"/> </rdfs:subClassOf> </owl:Class>
 <owl:Class rdf:ID="ConnectionStatus"> <rdfs:subClassOf rdf:resource="#DisconnectionStatus"/>  </owl:Class></rdf:RDF>
```

Figura 2. Un fragmento del archivo *Device Profile XML*

El *Mobile Device Agent (MDA)* se ejecuta en el *DM* del usuario. Su conocimiento está compuesto de reglas generales de comportamiento y características relacionadas con el tipo de *DM* usado (e.g., *PDA*), así como de algunos roles específicos definidos de acuerdo con la aplicación (e.g., este agente se usa para transmitir un archivo). Adicionalmente, un *MDA* debe conocer los protocolos de comunicación (conexión, tipo de red, etc.) compartidos con el sistema. El *MDA* maneja archivos *XML* (*Device Profile XML*, localizado en el *DM* del usuario, ver Figura 2) que describe las características del *DM* (utilizando *OWL - http://www.w3.org/2004/OWL/*) y comparte esta información con el *DisplayFilterAgent* (del *SMA* de *Adaptación*) a través del *Connection Controller Agent* (el *MDA* envía este archivo al *Connection Controller Agent* e intercambia esta información con el *DisplayFilterAgent*). El archivo *Device Profile XML* contiene cierta información acerca de los requerimientos de hardware y de software, características de la aplicación que se ejecuta durante una sesión, el tipo de archivos hipermedia que el *DM* soporta, las condiciones para desconexión (i.e. finalización de sesión): sesión inactiva por más de *X* minutos, tipo de desconexión (involuntaria, automática, etc.), etc.
Un *MDA* también maneja otro archivo *XML* que describe las características de la sesión del usuario (*Session XML*): quién está conectado, cuándo se inicio la sesión y cuál es el *DM* conectado. Este archivo se enviará al *UserAgent* (del *SMA* de *Adaptación),* al igual que el *User Profile XML,* archivo que contiene las preferencias del usuario.
El *Connection Controller Agent (CCA)* se ejecuta en la plataforma central de *PUMAS* y obtiene de una parte, la localización del usuario proveniente del archivo *User Location XML* (que contiene las características de la localización física y lógica del usuario) y, por otra parte, obtiene el tipo de *DM* (e.g., *PDA*) del archivo *Device Profile XML* (el cual contiene características de *DM*, ver Figura 2). Ambos archivos son provistos por los *MDAs* y localmente manejados por el *CCA*. El *CCA* sirve como intermediario entre los *SMA* de *Conexión* y el de *Comunicación.* El *CCA* también revisa las conexiones establecidas por los usuarios y el estado de los agentes (conectado, desconectado, eliminado, etc.). Adicionalmente, el *CCA* asocia cada *MDA* a su correspondiente *Proxy Agent* (agente que se ejecuta en el *SMA* de *Comunicación*).
Los archivos *XML* (*User Location, Session* y *Device Profile XML*) manejados por el *MDA* y el *CCA* son definidos usando las extensiones introducidas por Indulska *et al.* [4] a *CC/PP* [12]. Estas extensiones incluyen algunas características del usuario como su localización, perfil, requerimientos de las aplicaciones y características de la sesión.

2.2 El SMA de Comunicación

Este *SMA* está compuesto de uno o más *Proxy Agents*, un *MDProfile Agent* y un *Coordinator Agent*. Estos agentes se ejecutan en la plataforma central de *PUMAS*.

Hay un *Proxy Agent (PA)* por conexión de un *Mobile Device Agent (MDA)*. Dos usuarios diferentes pueden conectarse al sistema a través del mismo *DM* con lo que se tendrían dos diferentes *PAs* y dos sesiones diferentes. La principal actividad del *PA* es representar un *MDA* al interior del sistema. En este caso, hay dos agentes, un *Mobile Device Agent* que se ejecuta en el *DM* y un *Proxy Agent* que se ejecuta en la plataforma central de *PUMAS*. El *PA* tiene las mismas propiedades y comportamiento que el *Mobile Device Agent* excepto aquellas concernientes a la conexión.

El *MDProfileAgent (MDPA)* debe revisar el perfil de usuario (de acuerdo a su *DM*) y a sus necesidades de información. Adicionalmente, este agente junto con el *Coordinator Agent* define y revisa los mecanismos para enviar por ejemplo, datos hipermedia al usuario. Si los resultados de la consulta del usuario se traducen en varias imágenes, estos agentes definen el orden y el número de imágenes que serán mostradas a través de la pantalla del *DM* del usuario (de acuerdo a sus restricciones técnicas). El *MDPA* también comparte información con el *DisplayFilterAgent* (del *SMA* de *Adaptación*) sobre las características específicas del *DM* para la sesión del usuario.

El *Coordinator Agent (CA)* está en permanente comunicación con el *Connection Controller Agent* con el fin de verificar el estado de la conexión del agente que solicita la información. El *Coordinator Agent* conoce todos los agentes conectados en el sistema (utilizando un mecanismo de páginas amarillas) gracias a los archivos *XML* manejados por el *Mobile Device Agent* (a través del *Proxy Agent* que representa al *Mobile Device Agent*): sus conexiones, estado, servicios, localización, y características técnicas del *DM* del usuario. Si hay algún problema con el *Connection Controller Agent* (por ejemplo, si dicho agente falla o si hay más conexiones de las que puede manejar...), el *Coordinator Agent* puede desempeñar el rol de *Connection Controller Agent* hasta que se solucionen los problemas. En ese momento, el *Connection Controller Agent* y el *Coordinator Agent* deben sincronizar la información sobre los agentes conectados y el estado de sus conexiones actuales.

2.3 El SMA de Información

El *SMA* de *Información* está compuesto de un *Receptor/Provider Agent*, un *Router Agent* y uno o más *ISAgents*.

El *Receptor/Provider Agent (R/PA)* que se ejecuta en la plataforma central de *PUMAS,* tiene una visión general de todo el sistema. El *R/PA* recibe todas las consultas provenientes del *SMA* de *Comunicación* y las redirige al *Router Agent (RA),* agente encargado de encontrar el *SI* "*adecuado*" que ejecute la consulta. Una vez la consulta ha sido procesada por los *ISAgents* (que se ejecutan en los *SIWs*), el *R/PA* verifica si los resultados de la consulta toman en cuenta el perfil del usuario (información provista por el *ContentFilterAgent* del *SMA* de *Adaptación*). El *ContentFilterAgent* es responsable de actualizar las preferencias del usuario utilizando "*hechos*" (ver sección 4) almacenados en su *BC* y se comunica con el *UserAgent* (del *SMA* de *Adaptación*), agente que maneja la información del usuario para una determinada sesión.

Con el fin de redireccionar la consulta hacia los *SI* "*adecuados*", el *Router Agent (RA)* aplica una *estrategia* que depende de uno o más criterios (ver sección 6). El *RA* también está a cargo de recopilar los resultados provenientes de los *ISAgents* y de analizarlos (de acuerdo a los criterios definidos en las preferencias del usuario) para decidir qué parte de los resultados debe ser enviada al *R/PA*. Con el fin de enviar las consultas y analizar sus resultados, el *RA* debe revisar las preferencias del usuario (información provista por el *ContentFilterAgent* a través del *R/PA*).

Un *ISAgent (ISA)* asociado con un *SIW* (que se ejecuta en el mismo dispositivo del *SIW*) recibe del *RA* las consultas del usuario y es el encargado de buscar la información al interior del *SIW*. Cuando se obtiene el resultado de la consulta, el *ISA* lo envía al *RA*. Un *ISA* puede ejecutar una consulta o delegar esta tarea al componente adecuado del *SIW*. Esto depende de la naturaleza del *SIW*. Nuestra propuesta se dirige por una parte, a *SIW* complejos y posiblemente distribuídos, localizados en servidor(es) y por otra parte, a *SIW* simples consistentes en algunos archivos almacenados en un *DM*. En este último caso, un *ISA* es suficiente para asegurar el correcto funcionamiento del *SMA* de *Información*. Vale la pena anotar, que en este caso lo que llamamos un "*ISA*" es verdaderamente un *Mobile Device Agent* de un *DM* que puede jugar el rol de un *ISA* ya que cuenta con el conocimiento requerido para ejecutar la consulta, buscando información en los archivos almacenados en el *DM*. En un *SIW* complejo, el *ISA* puede colaborar con otro *ISA* (si el *SIW* ha sido desarrollado siguiendo el paradigma de *SMA*) o con otro componente del *SIW* para ejecutar la consulta. En el caso de un *SIW* no desarrollado como un *SMA*, nuestra propuesta sólo requiere que haya un *ISA* que se ejecute en dicho *SIW* y que pueda buscar información en su interior, garantizando la comunicación entre *PUMAS* y el *SIW*.

Debido a que la localización de un *SI* puede cambiar (especialmente si el *SI* se ejecuta en un *DM*), el *RA* puede ser informado sobre la localización actual del *SI* a través del *ISA* que se ejecuta en dicho *SI*.

2.4 El SMA de Adaptación

Las capacidades de adaptación de *PUMAS* se basan en un proceso de filtro compuesto de dos etapas cuyo objetivo es brindarle al usuario información adaptada a sus necesidades y tanto a sus características como a las de su *DM*. Primero, *el Filtro de Contenido* permite seleccionar la información más relevante de acuerdo al perfil de usuario definido en el sistema. Segundo, el *Filtro de Despliegue* se aplica a los resultados del primer filtro y toma en cuenta las características y restricciones técnicas del *DM* del usuario.

El *SMA* de *Adaptación* está compuesto de uno o más *UserAgents*, un *DisplayFilterAgent* y un *ContentFilterAgent*. Estos agentes se ejecutan en la plataforma central de *PUMAS*.
Cada *UserAgent (UA)* maneja un archivo *XML* (*User Profile XML*) que contiene las características del usuario (*ID*, localización, etc.) y sus preferencias (e.g., *"el usuario sólo desea información en video"*). Este archivo lo obtiene por medio del *Mobile Device Agent* (del *SMA* de *Conexión*). Sólo hay un *UA* por usuario en un momento determinado (incluso si el usuario tiene abierta dos sesiones en uno o diferentes *DMs*). Debido a que un usuario puede acceder al sistema a través de diferentes *DMs,* el *UA* se comunica con los *Mobile Device Agents* y los *Proxy Agents* (estos últimos pertenecientes al *SMA* de *Comunicación*) para analizar y centralizar todas las características del mismo usuario. El *UA* se comunica con el *ContentFilterAgent (CFA)* para enviar el archivo *XML*. Cuando el *CFA* recibe este archivo, almacena esta información como "*hechos*" en su *BC* (es decir, este agente almacena las preferencias de usuario). Cuando el *Receptor/Provider Agent* (del *SMA* de *Información*) solicita al *CFA* las preferencias del usuario, éste último le envía el archivo *XML* más reciente recibido del *UA*. Si el *UA* no le ha enviado este archivo (e.g., no hay preferencias específicas para la sesión actual), el *ContentFilterAgent* toma en cuenta para este usuario sus preferencias de sesiones anteriores.
El *DisplayFilterAgent* (*DFA*) maneja una *BC* que contiene información general acerca de las características de los diferentes tipos de *DMs* (e.g., los formatos de archivo que soporta) y el conocimiento adquirido de conexiones anteriores (e.g., problemas y capacidades de redes de acuerdo a la transmisión de datos). El *Connection Controller Agent* (del *SMA* de *Conexión*) se comunica con el *DFA* con el fin de solicitarle información sobre características y/o problemas de conexión (e.g., ancho de banda, velocidad de transmisión). El *MDProfile Agent* (del *SMA* de *Comunicación)* también se comunica con el *DFA* para solicitarle información sobre las capacidades y restricciones de un *DM* específico.
El *ContentFilterAgent (CFA)* maneja una *BC* que contiene las preferencias, intenciones y características de los usuarios. El *CFA* se comunica con el *UA,* solicitándole información sobre las preferencias de un usuario para una sesión específica (e.g., la sesión actual). Si el *UA* tiene preferencias específicas para una sesión, el *CFA* se las comunica al *Receptor/Provider Agent (R/PA, del SMA de Información)*. De otra manera, el *CFA* las busca (provenientes de sesiones anteriores) en su *BC*. Las características del usuario en el sistema son solicitadas por el *R/PA* que las adiciona a las consultas del usuario. Este agente verifica si los resultados son adaptados de acuerdo a esta información.

3. ESCENARIOS DE PUMAS

En esta sección, presentamos algunos escenarios que muestran cómo los agentes de *PUMAS* interactúan cuando un usuario ejecuta una consulta. En nuestra proposición, las interacciones de los agentes se basan en el intercambio de mensajes donde se utilizan los *Actos de Comunicación (Communication Acts)* presentados por Odell *et al.* en [7] (e.g., confirm, inform, propose, request, subscribe, propagate …).

3.1 Escenario de Conexión

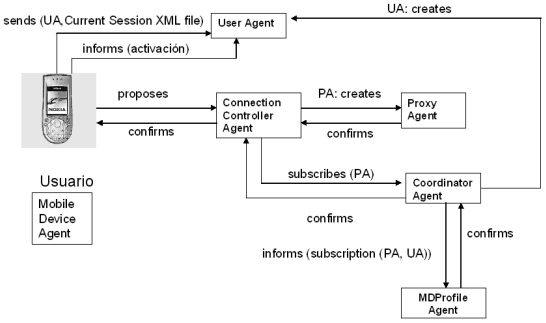


Figura 3. Escenario de conexión en PUMAS

Cuando un usuario se quiere conectar al sistema a través de su *DM*, el *Mobile Device Agent (MDA)* que se ejecuta en el *DM* del usuario envía un mensaje de "*propose"* (proposición de conexión) al *Connection Controller Agent (CCA)*. Si no hay un *Proxy Agent (PA)* que represente este *MDA*, el *CCA* crea uno y le envía un mensaje de "*subscribe*" al *Coordinator Agent* con el fin de suscribir este *Proxy Agent* al sistema. El *Coordinator Agent* informa al *MDProfile Agent* de esta suscripción. El *Connection Controller Agent* también envía al *MDA* un mensaje de "*confirms*" cuando el proceso de suscripción ha terminado (ver Figura 3). Cuando el *MDA* recibe el mensaje de confirmación, se crea un *UserAgent (UA)* en la plataforma central de *PUMAS* con el fin de manejar el perfil de usuario. Este perfil se define en el *archivo Session XML* y es enviado por el *Mobile Device Agent* al *UserAgent.*

3.2 Escenario de envío de una consulta

Cuando un usuario envía una consulta *C* (ver Figura 4), el *Mobile Device Agent* la envía al *Connection Controller Agent (CCA).* Cuando la consulta *C* depende de la localización y del tiempo de conexión, el *CCA* agrega esta información a la consulta *C,* así como las características de conexión del *DM* (esto último conocido a través del *DisplayFilterAgent*). Esta

nueva consulta *C'* (en Figura 4, *C'=C + localización*) es enviada al *Proxy Agent*. *C'* pasa por el *Coordinator Agent* y luego por el *MDProfile Agent*. Este último adiciona a la consulta *C'* algunas características relacionadas con el *DM;* estas características son provistas por el *DisplayFilterAgent* que las ha *aprendido* de consultas anteriores o las ha *recuperado* de su *BC*. La nueva consulta *C''* (en Figura 4, *C''= C' + características del DM*) es enviada por el *MDProfile Agent* al *Receptor/Provider Agent.* Este último agente adiciona a *C''* las características específicas del usuario que el *ContentFilterAgent* le ha dado (En Figura 4, *C'''=C'' + Preferencias del usuario*). El *Receptor/Provider Agent* envía la *C'''* al *Router Agent* que decide (de acuerdo a la consulta, el sistema de reglas y los hechos de su *BC*), qué *ISAgents* son capaces de responderla. El *Router Agent* puede enviar la consulta a un *ISAgent* específico o a varios (e.g., esperando el primero en responder o diferentes respuestas …) o, puede dividir la consulta en subconsultas que pueden ser enviadas a uno o más *ISAgents*. El escenario de la Figura 4 muestra cómo la consulta *C'''* se divide en $C'''^{-1.1}$, $C'''^{-1.2}$, $C'''^{-1.3}$ y $C'''^{-1.4}$ que son enviadas a los *ISAgents* que se ejecutan en un servidor y diferentes *DMs* (Las consultas son enviadas de un agente a otro utilizando el acto de comunicación *Request)*.

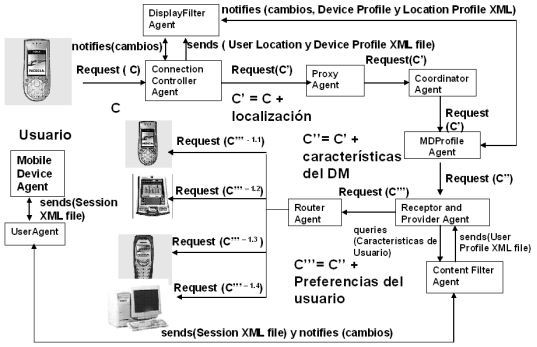


Figura 4. Escenario de envío de una consulta.

Cuando un usuario *U1* desea hacer una consulta a otro usuario (*U2*), ambos equipados con *DMs,* la consulta se propaga desde el *Mobile Device Agent (MDA1)* que se ejecuta en el *DM* de *U1* hasta el *Router Agent* que la redirige al *Mobile Device Agent (MDA2)* que se ejecuta en el *DM* de *U2*. El *MDA2* de *U2* cambia de rol y llega a ser un *ISAgent,* es decir, el agente a cargo de responder la consulta. Este cambio de rol es posible porque un *MDA* tiene el conocimiento para manejar la información almacenada en el *MD* en el cual se ejecuta y la capacidad de responder las consultas.

3.3 Escenario de recepción de los resultados de una consulta

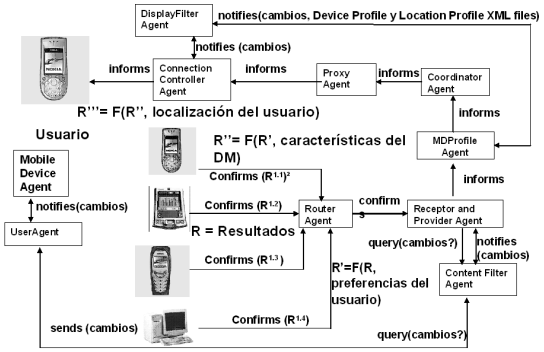


Figura 5. Escenario de recepción de los resultados de una consulta

Cuando el *Router Agent* recibe de los *ISAs* todos los resultados de la consulta (en la Figura 5, el *Router Agent* recibe los resultados parciales – $R^{1.1}$, $R^{1.2}$, $R^{1.3}$ y $R^{1.4}$ – enviados por los *ISAs* que se ejecutan en un servidor y diferentes *DM*s), los analiza antes de enviar mensajes de "*confirms*" o "*disconfirms*" o "*not understand*" al *Receptor/Provider Agent (R/PA).* Este mensaje incluye los resultados de la consulta (*R*). El *R/PA* revisa si los resultados satisfacen las características específicas del usuario (en la Figura 5, *R'* es el resultado de aplicar el *Filtro de Contenido* a *R* de acuerdo a las preferencias del usuario) y las redirige al *MDProfile Agent*. Este agente verifica si los resultados pueden ser desplegados de acuerdo a las características del *DM* y lleva a cabo el primer paso del *Filtro de Despliegue* (en la Figura 5, *R"* es el resultado de filtar *R'* de acuerdo a las características del *DM*). Luego, *R"* se transmite desde el *Coordinator Agent* hasta el *Proxy Agent.* Este lo envía finalmente hasta el *Connection Controller Agent* que lleva a cabo el último paso del *Filtro de Despliegue* de acuerdo a las características técnicas y de conexión del *DM* (si el usuario está aún conectado y ha cambiado de localización, o si ha ocurrido un *timeout*, etc.). Gracias a los filtros de *Contenido* y de *Despliegue*, los

resultados de la consulta recibidos por el *Mobile Device Agent* y desplegados en el *MD* del usuario corresponden a la información más relevante para el usuario.
Vale la pena anotar que el escenario descrito arriba incluye varias revisiones que podrían parecer inútiles ya que en el escenario de envío de una consulta se tienen en cuenta tanto las características del usuario como las de su *DM*. Sin embargo, esta información extra adicionada durante el proceso de envío de la consulta podría no ser válida en el momento en el que se entregan los resultados de dicha consulta. Las características del usuario (cambios de localización, de preferencias…) y los medios de comunicación y conexión (variación de ancho de banda, uso de diferentes *DMs*…) podrían haber cambiado y tener un impacto sobre los resultados esperados. Los controles que se presentan en este escenario se orientan a eliminar o reducir información que resulte ser irrelevante ante los cambios ocurridos.

4. MANEJO DEL CONOCIMIENTO EN PUMAS

En esta sección, describimos el conocimiento que es manejado por los agentes de los *SMA* de *Información* y de *Adaptación* de *PUMAS* para llevar a cabo su labor de adaptación de la información. Este conocimiento es almacenado en *Bases de Conocimiento (BC)* como *"piezas de conocimiento"*. Llamamos estas piezas "*hechos"* y las definimos usando *JESS* (*http://herzberg.ca.sandia.gov/jess/*). En la definición de dichas piezas utilizamos la sintaxis de *hechos no ordernados* de *JESS (unordered facts)*. Declaramos cada hecho mediante la primitiva "*deftemplate*"con el fin de representar las preferencias del usuario, las características del *DM*, los *SI*s, etc., como se describe en las siguientes subsecciones. Para definir una instancia de un *hecho* en *JESS* y luego insertarla en la *BC JESS* usamos la primitiva *"assert"*. Hemos elegido *JESS* debido a su compatibilidad con *JADE-LEAP (http://jade.tilab.com/)*, plataforma de *SMA* en la que se está desarrollando *PUMAS*.

4.1 Conocimiento del SMA de Información

El *Router Agent (RA)* almacena en su *BC,* un hecho por cada *Sistema de Información* (*SI)*. El *RA* los utiliza para redireccionar las consultas del usuario. Un hecho que representa un *SI* describe ciertas características como su nombre, la información que maneja, el tipo de dispositivo donde se ejecuta (por ejemplo, servidor, PC, *DM*) y el *ISAgent* asociado con el *SI* (es decir, el agente al que se le puede consultar la información y consecuentemente responde dichas consultas). Por ejemplo, el siguiente template *JESS* define un *SI*:

(deftemplate SI (slot nombre) (slot IDAgente) (slot dispositivo) (multislot info_items)) ; hecho (1)

El siguiente hecho (instancia del template definido anteriormente) representa el *SI* de una tienda. El *SI* es llamado *SIArticulos* y se ejecuta en un *servidor*. El *ISAgent* que se ejecuta en este *SI* se llama *ArticulosISA*. El *SIArticulos* contiene información (una lista de artículos - *info_items* -) entre los cuales se mencionan los artículos, las promociones y los nuevos productos que esta tienda vende:

(assert (SI (nombre SIArticulos) (IDAgente ArticulosISA) (dispositivo servidor) (info_items "articulos" "promociones" "nuevos productos")))

4.2 Conocimiento del SMA de Adaptación

Consideramos que las consultas pueden depender de uno o más criterios: la localización del usuario, su historia en el sistema, sus actividades desarrolladas durante un periodo de tiempo, sus preferencias, etc. Tales *Criterios de Dependencia* (*CriterioDependencia)* son definidos de la siguiente manera:

(deftemplate CriterioDependencia (slot usuarioID)(multislot criterios)(multislot atributos)) ; hecho (2)

El siguiente ejemplo expresa que todas las consultas de *John Smith* dependen de su localización, especialmente si El está en el aeropuerto:

(assert (CriterioDependencia (usuarioID "John Smith")(criterios localización)(atributos "aeropuerto")))

El *DisplayFilterAgent* maneja una *BC* que contiene información general sobre las características de los diferentes tipos de *DMs*. Cada *CaracterísticaDM* es definida como un hecho y se representa como sigue:

(deftemplate CaracterísticaDM (slot tipoDM) (multislot característica)) ; hecho (3)

Donde cada *característica* es definida a manera de *hecho* como se describe a continuación:

(deftemplate característica (slot tipo)(multislot condiciones)) ; hecho (4)

Las *condiciones* de una *característica* son aquellas que ésta necesita para ser satisfecha (e.g., el tipo de red que soporta algún tipo de dato). A continuación damos un ejemplo de una *característica* que corresponde al formato de archivos que soporta una *Palm Tungsten C* en diferentes tipos de red. Suponemos que este *DM* soporta video y varias imágenes cuando se conecta a través de una red *Wi-Fi*:

(deffacts característica (tipoDM "Palm Tungsten C")(característica (tipo video_soportado) (condiciones "red Wi-Fi"))
(característica (tipo varias_imágenes) (condiciones "red Wi-Fi"))) ;

El *ContentFilterAgent (CFA)* maneja una *BC* que contiene información acerca de las preferencias del usuario. Estas preferencias son representadas como hechos definidos de la siguiente manera:

(deftemplate Preferencia_Usuario (slot IDusuario)(slot info_requerida)(multislot info_complementaria)(multislot acciones_a_ejecutar)

(slot problema)(multislot acciones_para_recuperar))) ; hecho (5)

El hecho que representa una *Preferencia_Usuario* está compuesto del identificador de usuario propietario de esta preferencia (*IDusuario)*, la información requerida (*info_requerida*) y la información complementaria (info_complementaria). El *CFA* adiciona esta *info_complementaria* a la *Preferencia_Usuario*. El *CFA* también analiza las consultas efectuada en sesiones anteriores (información frecuentemente solicitada de manera simultánea con la *info_requerida*). Este hecho también se compone de la información acerca de cómo le gustaría al usuario que el sistema presente los resultados (lista de acciones a ejecutar para desplegar la información al usuario) y en caso de problemas, qué acciones ejecutar (*acciones_para_recuperar*). Para eso, cada *acción* se representa como sigue:

(deftemplate acción (slot nombre)(multislot atributo)) ; hecho (6)

En esta definición, *nombre* se refiere a una acción escogida de una lista definida (mostrar, salvar, transferir archivo, cancelar...). Cada *acción* tiene una lista de *atributos*. Por ejemplo, la acción "*mostrar*" que tiene como *atributos* el *orden,* el *formato* y el *tipo* de *archivo* que va a ser mostrado, es definida de la siguiente manera:

(assert (acción (nombre mostrar)(atributos "orden""formato""tipo_archivo")))

Debido a que un *atributo* puede ser complejo, lo definimos como un *hecho* de la siguiente manera:

(deftemplate atributo (slot nombre)(multislot lista)) ; hecho (7)

Un ejemplo de *atributo* que define el *orden* en el cual la información es desplegada en el *DM* es:

(assert (atributo (nombre orden)(lista "promociones" "nuevos_productos")))

Podemos definir un *problema* como algo inesperado o no deseado que ocurre cuando una acción se ejecuta (e.g., demora en la transmisión de datos), o, que es la consecuencia de la ejecución fallida de una acción (e.g., imposible mostrar una gran imagen a través de determinado *DM*). Cada *problema* es definido de la siguiente manera:

(deftemplate problema (slot nombre)(slot tipo)(multislot causas)) ; hecho (8)

Donde *nombre* corresponde a una descripción del problema, el *tipo* puede ser escogido de una lista definida (*incompatibilidad, agente no disponible*…), y las *causas* corresponden a una lista de posibles causas del problema (e.g., *DM* no puede soportar un formato de archivo específico). Por ejemplo, el siguiente hecho define el problema de que un *DM* no puede soportar algún tipo de información multimedia (e.g., un video). El problema definido abajo es considerado como una *incompatibilidad* causada por la reducida *memoria_DM* y por el reducido *tamaño_pantalla*:

(assert (problema (nombre videonosoportado) (tipo incompatibilidad)(causas "memoria_DM" "tamaño_pantalla")))

5. EJEMPLO

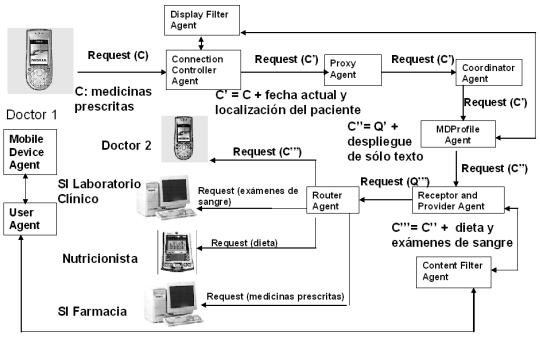


Figura 6. Envío de una consulta en el SIW de un hospital

En esta sección, ilustramos el proceso de consulta ejecutado por los agentes de *PUMAS* mediante un *SIW* de un hospital. Permítanos suponer que doctores equipados con *DMs* (e.g. *PDA*) acceden al *SIW* de un hospital. Dicho *SIW* puede estar distribuido entre diferentes *DMs* y/o uno o más *SIW* (ver Figura 6). Los doctores pueden también recibir información que concierne a sus pacientes de acuerdo a su localización, preferencias, características técnicas de sus *DMs* y consideraciones acerca del momento en el que se conectan. Por ejemplo, al visitar un paciente, los doctores equipados con *DMs* pueden consultar información sobre la historia clínica del paciente, sus análisis médicos, los medicamentos, etc. Para este ejemplo, mediante la localización del paciente (e.g., habitación) y la fecha actual, el doctor puede identificar al paciente y obtener su información. Para esto, la aplicación que se ejecuta en su *DM* debe consultar los diferentes *SI* del hospital – farmacia, pacientes, doctores, etc. Los doctores podrían también comunicarse con otros doctores (*peers*) a través de sus *DMs* mediante consultas específicas dirigidas a sus colegas (e.g., aquellas consultas que sólo puede responder el médico especialista que ha examinado recientemente al paciente).

Cuando un doctor ingresa la información concerniente a la localización del paciente y la aplicación toma la fecha del sistema, el *Mobile Device Agent* que se ejecuta en el *DM* del doctor envía la consulta (*¿Quién es el paciente?*). La consulta se propaga a través de la plataforma central de *PUMAS*: primero se transmite hacia el *Connection Controller*

Agent, luego a los agentes del *SMA* de *Comunicación* (*Proxy Agent, Coordinator Agent y MDProfile Agent*). El *MDProfile Agent* adiciona a la consulta, la información de acuerdo al *DM* (e.g., debido a que cierta clase de *DM* no soporta archivos gráficos, sólo texto, el doctor sólo va a obtener los resultados en formato texto). Por ejemplo, si el doctor se conectó a través de una *Palm Tungsten C,* el *MDProfile Agent* puede solicitar al *DisplayFilterAgent* información acerca de este *DM*. El *MDProfile Agent* podría recibir del *DisplayFilterAgent* hechos como el siguiente:

(deffacts Caracteristica (DMtipo "Palm Tungsten C")(característica (tipo video_no_soportado)(condiciones red_Wi-Fi))
(característica (tipo varias_imágenes)(condicions red_ Wi-Fi)(característica (tipo texto)(condiciones red_Wi-Fi red_clásica Bluetooth))) ;

Posteriormente, el *MDProfile Agent* envía la consulta al *Receptor/Provider Agent.* Este último agente le adiciona a la consulta las preferencias que el doctor ha expresado previamente. Las preferencias se encuentran registradas en el archivo *User Profile XML* (ver sección 2.4). Tanto el *UserAgent* como el *ContentFilterAgent* las traducen en hechos. El siguiente ejemplo corresponde a una preferencia de un doctor: "*al solicitar los resultados de los análisis de sangre de un paciente, el sistema también debe proveer la dieta y sus medicamentos prescritos; se prefiere obtener los resultados gráficamente; si el DM no soporta este formato, los resultados se recibirán en formato texto*". El *UserAgent* la traduce como el siguiente hecho:

(deffacts Usuario_Preferencia (IDusuario "Doctor Smith")(info_requerida "análisis de sangre")(info_complementaria "dieta del paciente" "medicamentos prescritos")(acción mostrar) (atributos (nombre orden) (lista "análisis del paciente" "dieta del paciente" "medicamentos prescritos"))(atributos (nombre formato_gráfico) (lista "JPEG"))(problema (nombre tipo_datos_no_soportado) (tipo incompatibilidad) (causas SoloTextoSuportado))(atributos (nombre orden) (lista "análisis del paciente" "dieta del paciente" "medicamentos prescritos"))
(atributos (nombre formato_texto) (lista "XML" "txt")))

El *UserAgent* transfiere esta información al *ContentFilterAgent* que almacena este hecho y lo envía al *Receptor/Provider Agent (R/PA).* El *R/PA* adiciona esta preferencia a la consulta y la envía al *Router Agent (RA)*. El *RA* recibe la consulta completa y con la información que éste tiene sobre los *ISs*, puede dividirla en subconsultas, redirigiendo cada una de ellas hacia el *SI* apropiado (ver sección 6). Los siguientes hechos son utilizados en este ejemplo por el *Router Agent* con el fin de redirigir las subconsultas a los *ISAgents* de los *SI* del hospital:

(assert (SI (nombre SILaboratorio) (IDAgente ISALab) (dispositivo servidor) (info_items "análisis de pacientes" "reactivos" "instrucciones")))
(assert (SI (nombre SIDietaPaciente) (IDAgente ISADieta) (dispositivoDM)(info_items "dieta de paciente" "citas del nutricionista")))

El *Router Agent (RA)* redirige la consulta al *ISAgent (ISA)* localizado en los *SIs* que manejan la información acerca de los pacientes en el hospital. Todas las consultas siguen el mismo recorrido desde el *Mobile Device Agent* hasta el *RA*. Si el doctor quiere conocer los últimos medicamentos prescritos a este paciente, el *RA* redirige la consulta al *ISA* localizado en el *SI* de la Farmacia. Si la consulta concierne a otro doctor, la redirige al *ISA* localizado en el *DM* de dicho doctor. Un doctor también puede solicitar información de un paciente específico a varios de sus colegas. En este caso, el *RA* podría enviar la consulta a manera de *broadcast* o podría dividirla de acuerdo al colega receptor (e.g., consultas relacionadas con el corazón al cardiólogo …) o a los criterios definidos en el archivo *User Profile XML* (e.g., si el criterio de dependencia de la consulta es la *localización*, las consultas deben ser redirigidas sólo a los doctores que se encuentran cerca del emisor). La información recuperada es organizada por el *RA* y se envía al emisor de la consulta siguiendo el recorrido inverso. Los diferentes agentes revisan los resultados porque, por ejemplo, el doctor puede haberse desconectado del sistema (debido a problemas de red), y recuperado su sesión a través de una nueva conexión cuyas características son diferentes de las anteriores: ahora el doctor podría consultar el sistema a través de otro *DM* que si soporta grandes imágenes (lo que haría que una de sus preferencias pueda ser ahora satisfecha).
A través de este ejemplo, podemos observar el comportamiento de *PUMAS* como un sistema *P2P híbrido*. El núcleo de *PUMAS* centraliza las consultas: está a cargo tanto del proceso de obtener la información más relevante como el de aplicar los filtros de *Contenido* y de *Despliegue* para adaptar las respuestas. Las principales características de un sistema *P2P* que poseen los agentes de *PUMAS* se ilustran en el hecho de que en primer lugar, los agentes tienen la autonomía de conectarse y desconectarse del sistema. En segundo lugar, un *DM* puede comunicarse con un *SI* específico pasando esta información como parámetro de la consulta; el *RA* transmite la consulta a este *SI* específico que ejemplifica una comunicación agente-agente (e.g., cuando los doctores intercambian información acerca de un paciente usando sus *DM*s). Otra ventaja que ofrece *PUMAS* es que el *RA* redirecciona la consulta mediante un análisis inteligente de la misma (ver sección 4) así como con la ayuda de los *ISAs,* agentes que llevan a cabo la búsqueda inteligente dentro de los diferentes *SI* (en nuestro ejemplo, los *SI* de farmacia, laboratorio clínico, pacientes, etc.).
En la siguiente sección, describimos el proceso de *Query Routing* que es especialmente llevado a cabo por el *Router Agent,* agente que aprovecha y utiliza el conocimiento que hemos descrito en la sección 4.

6. QUERY ROUTING EN PUMAS

El proceso de *Query Routing (QR)* en *PUMAS* es llevado a cabo por el *Router Agent (RA),* agente que recibe las consultas del usuario así como sus características y las de su *DM.* Con el fin de redirigir dichas consultas hacia los *SI*s "*adecuados*", la estrategia escogida por el *RA* depende de varios criterios: localización del usuario, similitud entre colegas, tiempo de conexión, etc. La estrategia puede estar orientada hacia el envío de la consulta a un *SIW* específico, o a su envío a manera de *broadcast*, o a dividir la consulta en subconsultas, cada una de ellas a ser enviada a uno o más *ISAgents* (agentes del *SMA* de *Información* que se ejecutan en el *SIW;* dichos agentes son los encargados de buscar la

información en el *SIW*). El *RA* también es el encargado de recopilar los resultados provenientes de los *ISAgents* y de analizarlos (de acuerdo a los criterios definidos para la consultas, ver sección 4.2) con el fin de decidir si todo el conjunto de resultados o sólo una parte de ellos será enviado al *Receptor/Provider Agent.*
En *PUMAS*, el proceso de *Query Routing* está compuesto de tres actividades (basadas en el trabajo de Xu *et al* [10]) que son descritas en las siguientes subsecciones. Explicamos cada actividad e ilustramos todo el proceso mediante el ejemplo de un *SIW* de un aeropuerto que describimos a continuación: Un pasajero equipado con su *DM* debe tomar un avión. Permítanos suponer que dicho pasajero debe llegar tres horas antes con el fin de registrarse y además para poder comprar algunos regalos en las tiendas del aeropuerto (*duty free shops*). Adicionalmente, cada aerolínea y cada tienda cuenta con un *SI* que provee a los usuarios con información acerca de sus servicios (e.g., el horario de las salidas y llegadas de vuelos) y sus productos (e.g., las promociones, novedades). A continuación presentamos algunos escenarios que serán usados a través de las siguientes subsecciones con el fin de mostrar cómo *PUMAS* ejecuta dicho proceso (*QR*)
Primer escenario: un pasajero desea conocer las tiendas (*duty free shops*) que venden los artículos de su lista de regalos. Suponga que varias tiendas venden el mismo producto (e.g., souvenirs, libros, postales, licores…), que corresponde a lo que el usuario quiere comprar. Además, dicho pasajero quiere saber qué tiendas son las *más cercanas* a la puerta de embarque de su vuelo y que venden cada artículo de su lista (al *precio más bajo*).
Segundo escenario: un pasajero desea consultar si su vuelo sale a tiempo.

6.1 Análisis de la Consulta
Esta actividad está relacionada con la posible división de la consulta. El *RA* analiza la complejidad de la consulta y decide si la consulta es *simple,* es decir, si sólo un agente la responde o, *compleja* si varios agentes son requeridos para responderla. Este análisis se basa en los hechos que el *RA* tiene almacenados en su *BC* acerca de los *SIs* (la información que maneja cada *SI*), los criterios de dependencia de la consulta (e.g., localización, similitud entre agentes), el conocimiento de los agentes que reciben las consultas (si la consulta está dirigida a un agente específico y conocido), etc. Después de este análisis, el *RA* decide si debe o no dividir la consulta en subconsultas.
Para el primer escenario, el *RA* debe dividir la consulta ("*todas las tiendas que venden los articulos de mi lista de regalos*") en varias subconsultas ("*todas las tiendas que venden un artículo específico de mi lista de regalos*"). El número de subconsultas depende del número de artículos. Si hay solamente un artículo, la consulta es *simple*, (sólo se enviará a un agente). De lo contrario, la consulta es *compleja*. Adicionalmente, el *RA* debe considerar dos criterios para la redirección de la consulta: la *proximidad* a la puerta de embarque y el *precio* del artículo en las tiendas. El *RA* debe además considerar las preferencias del usuario (ver hecho (5) y sus instancias, sección 4.2). Por ejemplo, el *RA* podría almacenar en su *BC* el siguiente hecho que expresa que cuando el pasajero *"John Smith"* consulta las *"tiendas más cercanas"* a una posición dada*,* El también solicita todas las tiendas que tienen los *"más bajos precios"*:

(assert (Preferencia_Usuario (IDusuario "John Smith") (info_requerida "tiendas más cercanas")(info_complementaria "más bajos precios") (acciones_a_ejecutar mostrar) (problema "lista vacía de tiendas")(acciones_para_recuperar cancelar)))

Para el segundo escenario, el *RA* no debe dividir la consulta ya que es simple y un agente de los *SIW* consultados podría responderla (el *ISAgent* de la aerolínea o el del aeropuerto que maneja las llegadas y salidas de los vuelos).

6.2 Selección de las Fuentes de Información
Una consulta puede ser dirigida a un agente específico o a un grupo de agentes; si la consulta debe ser dirigida a agentes conocidos (e.g., receptores específicos seleccionados por el usuario), la selección de las fuentes es simple (las fuentes de información son los agentes específicos). De lo contrario, el *RA* selecciona las fuentes de información y crea la "*red de vecinos*" (basado en las ideas de Yang *et al.* [11]). En nuestra proposición, un agente es vecino de otro si satisface un conjunto de características (definidas en las preferencias de usuario de la aplicación). Por ejemplo, la proximidad en la localización, las mismas actividades, conocimiento similar, colegas que trabajan en un mismo grupo, etc. Las características no se restringen a la proximidad en la localización. Para conformar la "*red de vecinos"* en la cual cada nodo es una fuente de información (agente), consideramos los siguientes casos:
Primero, varios agentes que pueden responder la misma consulta. La forma más simple de conformar la red es agrupar todos estos agentes. Este agrupamiento es útil, por ejemplo, cuando el *RA* no tiene información acerca de las fuentes o cuando es la primera vez que el *RA* trabaja con los potenciales vecinos. Con el fin de reducir comunicaciones que puedan resultar redundantes, innecesarias o inútiles y, de seleccionar los vecinos más relevantes, el *RA* aplica los criterios de dependencia para dicha consulta. Por ejemplo, si el criterio de dependencia es la *localización*, la red de vecinos será conformada por *los vecinos más cercanos*; si depende de *anteriores consultas*, el *RA* debe redireccionarla a *los vecinos más confiables* (aquellos que siempre han respondido de una manera adecuada, o más rápidamente, etc.); si el criterio de dependencia es la *similitud*, la red será conformada por los vecinos con un *perfil similar*, o que desempeñan *tareas similares*, etc. Si no hay criterio de dependencia establecido, el *RA* analiza el nivel de confianza de estos vecinos. Dicho agente asocia a cada vecino un nivel de confianza de acuerdo a sus anteriores interacciones con dichos vecinos; esta idea de confianza en los vecinos se basa en el trabajo de Agostini *et al.* [1].
Segundo, un sólo agente puede responder la consulta y éste es conocido. El *RA* utiliza el conocimiento que tiene almacenado en su *BC* sobre los *SIs* para contactar aquel agente que podría responderla.

Tercero, la consulta ha sido dividida en varias subconsultas en la etapa de análisis. El *RA* analiza qué agentes podrían responder cada una de las subconsultas y con ellos compone la red de vecinos. El proceso aplicado para seleccionar las fuentes de información para cada subconsulta es el mismo definido en el primer caso. Finalmente, la red de vecinos está compuesta de la unión de las diferentes subredes generadas para cada subconsulta.
Para el primer escenario, la red de vecinos contiene los *ISAgents* de las tiendas que venden los productos que el pasajero busca (ver hecho (1) y sus instancias, sección 4.1) y que podrían responder cada una de las subconsultas: "*las tiendas más cercanas a la puerta de embarque que venden el articulo requerido al más bajo costo*". Por ejemplo, en la *Base de Conocimiento* del *RA* habría hechos como el siguiente que define el *SI* de la tienda de licores del aeropuerto:

(assert (SI (nombre SITiendaLicores)(IDAgente TLicoresISA)(dispositivo servidor)(info_items "licores" "promociones" "nuevas marcas")))

Además, el *RA* aplica los criterios de dependencia para dicha consulta (ver hecho (2), sección 4.2), en este caso, la proximidad de las tiendas a la puerta de embaque y el precio más bajo de los artículos. Por ejemplo, el *RA* podría almacenar hechos en su *BC* como los presentados abajo. Dichos hechos expresan que todas las consultas de "*John Smith*" dependen tanto de la *localización* (si El está en el aeropuerto), como de la *proximidad* a la puerta de embarque:

(assert (CriterioDependencia (usuarioID "John Smith")(criterios localización)(atributos "aeropuerto")))
(assert (CriterioDependencia (usuarioID "John Smith")(criterios proximidad)(atributos "puerta embarque")))

El *RA* también analiza el nivel de confianza asociado a estos vecinos (e.g., la primera tienda en responder). Si es la primera vez que el *RA* ejecuta esta consulta o que trabaja con estos *ISAgent*s, el *RA* les envía la consulta a través de un mensaje transmitido a manera de *broadcast*.
Para el segundo escenario, el pasajero solicita información acerca de los horarios de los vuelos a los *SIWs* de la aerolínea y/o del aeropuerto (el *RA* envía la consulta a los *ISAgents* que se ejecutan en esos *SIs*).

6.3 Redirección de la consulta

Una vez el *RA* ha identificado las fuentes de información potenciales (vecinos), redirecciona la consulta mediante el envío de un mensaje a sus vecinos (que incluye la consulta). El *RA* puede usar un mensaje orientado (para receptores específicos) o un mensaje *broadcast* a todos los vecinos (e.g., esperando el primero en responder, u obteniendo todas las respuestas y luego analizando cuáles de ellas son más confiables, etc.). Si el *RA* contempla un esquema de confianza para los agentes que componen la red de vecinos, este agente podría enviar el mensaje secuencialmente, comenzando por el vecino más confiable. Si este responde, el proceso se termina. De lo contrario, el *RA* debe continuar el envío de mensajes hasta que haya contactado al agente menos confiable (de acuerdo a las ideas de Agostini *et al* [1]).
Para el primer escenario, el *RA* envía la consulta a los *ISAgents* que conforman la red de vecinos. Si el *RA* sabe que el *vecino1* puede responder la *subconsulta1*, que el *vecino2* la *subconsulta2* y así sucesivamente, envía un mensaje orientado a cada uno de los vecinos (basado en el hecho (1) y sus instancias; ver sección 4.1). Por ejemplo, para el segundo escenario, el *RA* envía la consulta al *ISA* que se ejecuta en el *SIW* de la aerolínea y/o al *ISA* que se ejecuta en el *SIW* del aeropuerto que maneja el horario de llegadas y salidas de vuelos. En este caso, podríamos encontrar en la *BC* del *RA,* los siguientes hechos que permitirían redireccionar la consulta a los agentes *OneISA* y al *SLLVuelosISA*:

(assert (SI (nombre AerolineaUnoSI) (IDAgente OneISA)(dispositivo servidor)(info_items "salidas" "llegadas" "vuelos" "tarifas")))
(assert (SI (nombre AirportIS)(IDAgente SLLVuelosISA)(dispositivo servidor)(info_items "salidas" "llegadas")))

Finalmente, el *RA* debe recopilar las respuestas obtenidas de los diferentes agentes y seleccionar las más relevantes de acuerdo a los criterios establecidos (como fue explicado en la sección 6.2).

7. TRABAJOS RELACIONADOS

En esta sección presentamos algunos de los trabajos cercanos al nuestro que corresponden a arquitecturas o frameworks basados en agentes que adaptan la información a los usuarios:
CONSORTS [6] es una arquitectura de agentes ubícuos diseñada para soporte masivo de *DMs.* Dicha arquitectura detecta la localización del usuario y define su perfil a través de un *Spatio-Temporal Reasoner* (*STR)*. *CONSORTS* utiliza la localización del usuario y su perfil para adaptarle la información. Además, propone mecanismos para definir las relaciones entre agentes (comunicaciones, jerarquía, definición de roles …), con el propósito de satisfacer las necesidades de información del usuario. Sin embargo, *CONSORTS* no considera aspectos como la distribución de información entre diferentes dispositivos (que podría mejorar tiempos de respuesta) ni la adaptación de la información a los dispositivos de acceso (*DMs*). Tampoco considera las preferencias del usuario.
El sistema *PIA* [2] es un sistema de información personal basado en agentes que recolecta, filtra e integra información en un punto comun, ofreciendo acceso a la información a través de WWW, e-mail, SMS, MMS y clientes J2ME. Combina técnicas de *push* y *pull* que permiten al usuario, por una parte, buscar explícitamente información específica y por otra parte, ser informado automáticamente sobre información relevante dividida en grupos (el usuario especifica su tiempo de trabajo y el sistema divide el día en pre, trabajo y recreación). Un agente personal maneja la información individual de cada usuario, brindándole información adaptada a su perfil. Cuenta con un mecanismo de retroalimentación para el perfil de usuario. Sin embargo, el sistema *PIA* solamente busca información en formato texto (es decir, documentos). Tampoco toma en cuenta la adaptación de la información a diferentes tipos de *DMs* ni la localización del usuario.

8. Conclusiones y Trabajo Futuro

En este artículo, hemos descrito el conocimiento manejado e intercambiado por los agentes de los *Sistemas Multi-Agente (SMAs)* de *Información* y de *Adaptación* de *PUMAS* con el fin de llevar a cabo la adaptación de la información. *PUMAS* es un framework que recupera a partir de diferentes *Sistemas de Información Web (SIW)*, información adaptada al perfil de usuario – que contiene sus necesidades, características, preferencias, historia en el sistema, localización actual, etc. – y, a las capacidades técnicas del *Dispositivo Móvil (DM)* que dicho usuario utiliza para acceder los *SIW*. También hemos descrito las estrategias seguidas por el *Router Agent* con el fin de llevar a cabo el proceso de enrutamiento de consultas (*Query Routing)*. En *PUMAS*, dicho proceso se compone de tres actividades: el *análisis de la consulta*, la *selección de fuentes de información* y la *redirección de la consulta*. Presentamos cada una de estas actividades y las ilustramos a través de diferentes escenarios de un *SIW* de un aeropuerto. Finalmente, mostramos algunos trabajos consistentes en *frameworks* o arquitecturas basados en agentes que adaptan la información al usuario.

Nuestro trabajo apunta a la continuación en la implementación de cada uno de los *SMA* de *PUMAS*. Para esto, hemos escogido *JADE-LEAP (http://jade.tilab.com/)*, una plataforma acorde con los estándares *FIPA*. Hemos ya probado los ejemplos que vienen con esta plataforma en dos tipos diferentes de *DMs* de nuestro equipo de investigación (*Pocket PCs* con *Windows CE*, que utiliza *Crème* – kVM, acorde con la implementación *Personal Java* - y *PDAs* con *PalmOS* utilizando una implementación *MIDP* 1.0). Los resultados de estas pruebas son presentados en [3].

Agradecimientos. *Angela Carrillo Ramos* es parcialmente financiada por la *Universidad de los Andes* (Colombia).

Referencias

[1] Agostini, A. and Moro, G. Identification of Communities of Peers by Trust and Reputation. *Proc. of the 11th Int. Conf. in Artificial Intelligence: Methodology, Systems, and Applications (AIMSA 2004)* (Varna, Bulgaria, September 2-4, 2004). LNCS, Vol. 3192. Springer-Verlag, Berlin Heidelberg, (2004), pp. 85-95.

[2] Albayrak, S., Wollny, S., Varone, N., Lommatzsch, A. and Milosevic D. Agent Technology for Personalized Information Filtering: The PIA-System. *Proc. of 20th Annual ACM Symposium on Applied Computing (SAC 2005)* (Santafe, New Mexico, USA, March 13-17, 2005). ACM Press, NY, NY (2005), pp. 54-59.

[3] Carrillo-Ramos, A., Gensel, J., Villanova-Oliver, M. and Martin, H. Modelling with Ubiquitous Agents a Web-based Information System Accessed through Mobile Devices. *Proc. of the Cooperative Inf. Systems (CoopIS'04).* (Larnaca, Cyprus, Oct. 25-29, 2004). LNCS, Vol. 3290, Springer-Verlag, Berlin Heidelberg (2004), pp. 264-282.

[4] Indulska, J., Robinson, R., Rakotonirainy, A., and Henricksen, K. Experiences in Using CC/PP in Context-Aware Systems. *Proc of 4th Int. Conference on Mobile Data Management (MDM 2003)* (Melbourne, Australia, January 21-24, 2003), LNCS, Vol. 2574. Springer-Verlag, Berlin Heidelberg (2003), pp. 247-261.

[5] Koch, F. and Rahwan, I. Classification of Agent-based Mobile Assistant. *Proc. of Workshop on Agents for Ubiquitous Computing (UbiAgents04)* (NY, USA July 20, 2004). *http://www.ift.ulaval.ca/~mellouli/ubiagents04/.*

[6] Kurumatani, K. Mass User Support by Social Coordination among Citizen in a Real Environment. *Proc. of Multi-Agent for Mass User Support. International Workshop (MAMUS 2003).* (Acapulco, Mexico, August 10, 2003). LNAI, Vol. 3012, Springer-Verlag, Berlin Heidelberg (2003), pp. 1–16.

[7] Odell, J., Van Dyke Parunak, H. and Bauer, B. Representing Agent Interaction Protocols in UML. *Proc of 1st Int. Workshop on Agent Oriented Software Engineering (AOSE 2000)* (Limerick, Ireland, June 10, 2000), LNCS, Vol. 1957. Springer-Verlag Berlin Heidelberg, (2000), pp.121-140.

[8] Park, J. and Barber, S. Finding Information Sources by Model Sharing in Open Multi-Agent System. *Proc. of Workshop on Agents for Ubiquitous Computing (UbiAgents04)* (NY, USA, July 20, 2004) *http://www.ift.ulaval.ca/~mellouli/ubiagents04/* (Ultima revisión: Marzo 2005)

[9] Pirker, M., Berger M. and Watzke, M. An approach for FIPA Agent Service Discovery in Mobile Ad Hoc Environments. *Proc. of Workshop on Agents for Ubiquitous Computing (UbiAgents04)* (NY, USA July 20, 2004). *http://www.ift.ulaval.ca/~mellouli/ubiagents04/* (Ultima revisión: Marzo 2005)

[10] Xu, J., Lim, E. and Ng, W.K. Cluster-Based Database Selection Techniques for Routing Bibliographic Queries. *Proc. of the 10th Int. Conf. and Workshop on Database and Expert Systems Applications (DEXA 99)* (Florence, Italy, August 30-September 3, 1999), LNCS, Vol. 1677. Springer-Verlag, Berlin Heidelberg, pp. 100-109.

[11] Yang, D., Xu, L., Cai, W., Zhou, S., and Zhou, A. Efficient Query Routing for XML Documents Retrieval in Unstructured Peer to Peer Networks. *Proc. of the 6th Asia Pacific Web Conference (APWeb 2004)* (Hangzhou, China, April 14-17, 2004). LNCS, Vol. 3007, Springer-Verlag, Berlin Heidelberg, (2004), pp.217-223.

[12] *http://www.w3.org/TR/webont-req/* (Ultima revisión: Marzo 2005)

A Fault-Tolerant Home-Based Naming Service For Mobile Agents

Camron Tolman and Carlos A. Varela
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
{*tolmac,cvarela*}*@cs.rpi.edu*

Abstract

A *naming service* is in charge of locating a mobile agent in a distributed system given its name. Three critical characteristics of a naming service are: *fault tolerance*, *scalability*, and *efficient name resolution*. Most naming services provide support for efficient name resolution but do not address fault tolerance or scalability issues. Conversely, distributed hash-table approaches to naming provide fault tolerance and scalability albeit at a sacrificed name resolution performance. We introduce a Fault-Tolerant Home-Based Naming Service (FHNS) that is robust and scalable yet enabling efficient name resolution.

Keywords: Mobile agents, fault-tolerance, naming services, universal actors

1 INTRODUCTION

A *naming service* maps a name for an entity to a set of labelled properties. The Domain Name System (DNS) [11], for instance, is a network naming service that maps *easy-to-remember* names to *hard-to-remember* network addresses. Similarly, name services can be put to use in distributed computing architectures to map the identity of a mobile software agent to its current location on the network. The Fault-Tolerant Home-Based Naming Service (FHNS) we introduce, is one such naming service.

The motivation to develop a new naming service comes from considering the under-utilization of the Internet, which is predominately used for information retrieval, correspondence, commerce, and file sharing. The Internet, which can be thought of as a super-computer consisting of over a million processors, offers a vast amount of computing capability that has yet to be fully exercised. Substantial computations, such as data mining, financial forecasting, and complex simulations, can be divided and distributed across the Internet to be serviced by numerous machines concurrently. In order for these computations to be distributed, a framework first needs to be structured and deployed. The framework is required to be efficient, robust, scalable, and secure. The purpose of this research, however, is not to present a distributed computing framework in its entirety but to present a naming service that enables such a framework to meet a measure of the identified requirements.

The requirements for a naming service intended for a network like the Internet are equivalent to those placed on a distributed computing framework. The naming service must be:

- **Efficient:** The messages associated with resolving the location of an agent should be many times less than the number of nodes on the network. Not every node can be queried when an agent is being located.
- **Robust:** Any one failure cannot result in the entire service failing. The naming service must be resilient to node and link failures.
- **Scalable:** The service should allow nodes and agents to be located anywhere in the world and be able to support a huge number of these entities.
- **Secure:** The service must ensure data integrity and thus protect the names it is entrusted with from unauthorized modification or deletion.

And the names for the mobile agents must be:

- **Unique:** This is a requirement common to all naming services. In this case, the requirement for uniqueness is even more extensive in that the names for the mobile agents are to be globally unique.
- **Persistent:** The names must not change unless the naming service is notified.
- **Human Readable:** This is a requirement we place upon ourselves with the notion that human readable names are easier to remember and use. A human readable name can also indicate the purpose the agent serves, as would be the case of naming a personal-calendar agent: *JoeDoesCalendar*. Furthermore, people have the possibility to guess the agent name.

Mobile agents are best described as software processes that can migrate from one host to another while executing their assigned tasks. In between migrations, a mobile agent may need to interact with other agents to complete its jobs. Say for instance, a protein folding application decomposes the computations associated with protein folding into sub-computations to be processed by mobile software agents. These software agents are dispersed throughout the Internet to perform their distributed tasks. After performing a number of its computations, software agent GYPSY needs software agent NOMAD's results in order to continue with its computations. How does GYPSY locate the whereabouts of NOMAD given that it could be at any node on the Internet? There are different possible strategies to address the problem of locating an agent for the purpose of communicating directly with that agent [14, 17, 5]: for example, broadcast, search, tracking, centralized dedicated name services, and distributed dedicated name services.

Structure of the Paper The remainder of the paper is organized as follows. Section 2 presents the model of the Fault-Tolerant Home-Based Naming Service. Section 3 presents an FHNS implementation that was tested. The results of these tests are presented and evaluated in Section 4. And lastly, Section 5 presents some concluding remarks and contributions of this research.

2 FAULT-TOLERANT HOME-BASED NAMING SERVICE(FHNS)

2.1 Model

The Fault-Tolerant Home-Based Naming Service conforms to the home base naming service construct and as such each mobile agent has a home base that keeps track of that agents location. The mobile agents can either specify their home base directly or can opt to have a home base assigned. The naming service recognizes and is capable of resolving two types of names: location-dependent names or location-independent names. Due to this versatility, the service can effectively shift from a location-dependent service to a location-independent service in the event of a node failure (i.e. home base failure).

FHNS is succinctly comprised of the following:

1. **Unique Names:** Mobile agents have unique names that are premised upon Uniform Resource Identifiers (URI) [3]. The URI can take the form of a URL [2], which is location-dependent and is how the agent specifies its home base, or the URI can take the form of a URN [10], which is location-independent and requires a home base to be assigned.
2. **Home Bases:** Nodes on the network that provide the naming service for the mobile agents.
3. **Mapping Records:** A record kept for a mobile agent that is kept current with the agent's location on the network. The record is stored at the agent's respective home base.

Table 1 details the API for the Fault-Tolerant Home-Based Naming Service, which are the specific home base services.

2.2 Names

The location-dependent naming is attractive for the obvious reason that the location of the home base is given explicitly. Consequently, the agents themselves can be found quickly and efficiently simply by querying the respective home base. The location-dependent names of an agent take the form of a URL where the general syntax is:

<location-dependent name> ::= <scheme>:<scheme-specific-part>.

Function	**Description**
`put`(name, location)	Binds a name to an initial location.
`get`(name)	Returns the location for the given name.
`update`(name, location)	Updates the location of the agent with the specified name.
`delete`(name)	Remove the location record for the agent with the specified name.

Table 1: Fundamental API for FHNS

A URL contains the name of the *scheme* being used, which is an existing or experimental protocol. The scheme is followed by a colon and then a string (the <scheme-specific-part>) whose interpretation depends on the scheme. For Internet schemes the scheme specific part is written as follows:

//<user>:<password>@<host>:<port>/<url path>.

The optional elements are "<user>:<password>@", ":<password>" and ":<port>". The scheme specific data begins with a double slash "//" to indicate that it complies with the common Internet scheme syntax. The *host* field corresponds to the domain name of the node, or home base for the purposes of FHNS. The *url path* is not an optional element for FHNS and corresponds to the relative name of the agent. The names for an agent are referred to as either relative or absolute. The absolute name refers to the name in its entirety. The relative portion of the name refers to the name given to the agent that distinguishes it from other agents using the same host as a home base.

Location-independent naming is appealing because it allows the home base to change for a given agent. With location-independent naming, the names of an agent can take the form of a URN. The URN syntax is:

"urn:" <NID> ":" <NSS>.

The *NID* is the *Namespace Identifier*, and *NSS* is the *Namespace Specific String*. A ":" delineates the three elements. The "urn" descriptor string is specified to be a required element; however, for the purposes of FHNS this string will be replaced by another string corresponding to an existing or experimental protocol[1]. This modification allows for the two types of names to share a common element. The FHNS modified URN will be written as

<location-independent name> ::= <scheme> ":" <NID> ":" <NSS>.

A URN differs from a URL in that its primary purpose is persistent labelling of a resource with an identifier as opposed to serving as a locator. The location of agents are then resolved by using *consistent hashing* [7] and a lookup service such as the Chord [16, 4] peer-to-peer lookup service. By using a peer-to-peer lookup service, the home bases are required to keep a listing of other home bases so that when a *lookup* is required a home base will be able to query other home bases for the information it needs, in this case the location of an agent. A lookup is the process of a home base querying other home bases and could be performed when servicing any of the naming service requests. A lookup is not synonymous with the `get` service request.

2.3 Architecture

FHNS can be described as a layered architecture with DNS serving as an underlying naming service. DNS is utilized by FHNS when resolving the domain name of a home base to an IP address.

Names in the Domain Name System are a part of a naming hierarchy, with the most significant part on the right. The left-most portion of the name corresponds to the name given to the computer. Clients query local DNS servers for IP addresses. If a mapping has not been locally cached, the server starts with the root name server and recursively queries DNS servers until it finds a server that has the answer. In the case of a client request regarding *www.yahoo.com*, the local server queries a root DNS server then a *com* DNS server, and then finally the *yahoo.com* server for the web server IP address.

Fault tolerance is defined as the ability of a system to respond gracefully to an unexpected hardware or software failure. Fault tolerance for DNS is achieved with the use of a designated primary server and a

[1]The names adhere to the rules specified in [3, 2, 10] with this exception being made to the names following the URN format.

secondary or "back-up" server (or servers). If the primary server fails, requests are directed to the secondary server. With this approach, however, the notion of *failover* is always provided for the primary but not the secondary. If the secondary server fails or the last "backup" server fails, there exists a fault in the system. FHNS achieves fault tolerance with redundancy between home bases and provides for an iterative failover capability. Each home base functions as both a primary and secondary server. And each home base has a designated successor that is capable of servicing requests on behalf of that home base in the event of a failure. Essentially, FHNS is capable of recovering from all home bases, except any arbitrary one, failing.

Returning our attention back to DNS, when an organization wants to participate in the Domain Name System, the organization must apply for a name under one of the top-level domains (i.e. com, edu, gov, co, fr, etc.). If the name is already in use, the organization will not be allowed to use the name and thereby uniqueness is ensured among all the top-level domain names. If the name is not already in use and the organization is allowed to use the name. They then assume the responsibility of ensuring that names are unique with respect to their domain. Names are kept unique in FHNS in a similar manner. In the case of location-dependent naming the first part of the name, that being the host, must be unique since it is a name in the Domain Name System. The name of the agent must then be unique with respect to the home base (i.e. host). The home bases reject a put request if the name requesting to be used is already in use—thereby ensuring uniqueness. Ensuring unique naming with the location-independent names requires that the combination of namespace identifier and namespace specific string be unique. Again the home bases can reject a put request if the namespace specific string is not unique relative to the namespace identifier. A single global namespace identifier will be used, which is thus trivially unique.

The namespace identifier corresponds to a logical ring with distinct points along the ring called *identifiers*. A single home base is assigned to one of the identifiers by way of consistent hashing. Consistent hashing is a distribution scheme that pertains to using a hash function to distribute strings to a number (i.e. identifier) in the range $[0,\ldots, M]$, where M is determined by the selected hashing function. Home Bases are assigned to an identifier by hashing the domain name or the IP address. The URN of an agent is also hashed and mapped to an identifier. The agent is then assigned a home base by identifying the home base that has been mapped to a number greater than or equal to the number the agent has been mapped to (0 is the first number greater than M, when making assignments). All home bases belong to this single logical ring and each home base keeps a routing table that lists other identifiers and the home bases associated with those identifiers. The routing table contains $log\ M$ entries[2]. Each home base has its mapping records replicated at a number of home bases with identifiers greater than and nearest to the identifier of the respective home base. The logical ring is not static, new home bases can join and the mapping records will be transferred accordingly. Conversely, home bases can leave the logical ring and one of the replicating home bases will assume responsibility for the departed home base's mapping records. However, the hosts that will serve as home bases should be chosen judiciously since it would not be desirable to have a home base that is only powered on for short durations, as would be the case with mobile devices. Also, home bases are not restricted to being dedicated name servers only. Home bases can be like other hosts in providing a platform for mobile agents to execute their assigned tasks.

2.4 Protocol

As an example of how the naming service functions, we now present a description of agents using the service. We begin with a 3-bit namespace, called $drifters$, and we will use *SHA-1 % 8* [21] as our hashing function. We identify 3 hosts to be our home bases and assign them each an identifier.

NAME	IDENTIFIER
rome:3030	7
hongkong:4040	0
newyork:5050	5

Table 2: Example: Home Bases and the Assigned Identifiers

In practice we would use a hashing function bigger than 3-bits to avoid collisions and identifiers would not be assigned but determined by hashing the domain name. Next we introduce 4 mobile agents, half

[2]Unless stated otherwise, all logarithms in this paper are of base 2.

will use location-dependent names and the other half of agents will use location-independent names. Our experimental naming service scheme is called *fhns*.

RELATIVE NAME	ABSOLUTE NAME	IDENTIFIER
MIGRANT	fhns://rome:3030/MIGRANT	6
TUMBLEWEED	fhns://hongkong:4040/TUMBLEWEED	0
GYPSY	fhns:drifters:GYPSY	4
NOMAD	fhns:drifters:NOMAD	2

Table 3: Example: The Agents' Names and the Determined Identifiers

The agents would need to be aware of at least one home base so that they could issue a request to that home base to perform a put(<agent absolute name>, <agent location>). Agents TUMBLEWEED and MIGRANT could issue the request to the host specified in their names, which would then become their corresponding home base. GYPSY and NOMAD could issue their requests to any of the home bases. The home base servicing the request first hashes the names of these agents, GYPSY and NOMAD, to determine the proper home bases and then forwards the request to those identified home bases. Home bases are able to forward requests by determining from their routing tables the host nearest the identifier they are seeking.

If GYPSY wanted to communicate with MIGRANT it could request that the *rome* host machine perform a `get`(*fhns://rome:3030/MIGRANT*) operation. The location of MIGRANT would be provided with a single message loop (i.e. request and reply), which is why we say that the location-dependent naming aspect of this naming service is very efficient in resolving actor locations. But say GYPSY wanted to communicate with NOMAD. GYPSY would direct a request to any of the home bases to perform a `get`(*fhns:drifters:NOMAD*) operation. The home base, say *hongkong*, would first hash the string "NOMAD", which is the agent's relative name, according to the *SHA % 8* hashing function associated with the *drifters* namespace. The relative name of NOMAD hashes to identifier 2. *Hongkong* would look into its routing table and see that *newyork* maps to identifier 5, which is the node greater-than and closest to identifier 2. The `get`(*fhns:drifters:NOMAD*) request would be handled by *newyork* and the results would be forwarded to *hongkong* and then provided to GYPSY. Here it took more than a single message loop to find the location of NOMAD. Location-independent naming does not give as good results in resolving the location of agents since it requires that the home base for an agent first be discovered. Finding the location of a home base could take $O(log\ n)$ messages when using the Chord lookup strategy, where n is the number of nodes in the ring (i.e. home bases). When a home base is requested to determine the location of an agent, it first checks the mapping records in its possession. If a mapping record is found, it returns the location of the agent. If a mapping record is not found, then the home base forwards the request to the home base that is closest to the identifier to which the name of the agent is mapped. The request is forwarded until it is serviced by the authorized home base. To minimize the number of hops between nodes, the searching home base can cache the location of an agent's home base so as to avoid having to perform another multi-message lookup for subsequent locate requests for the same agent.

Some time has passed and NOMAD decides to move from a host machine called *host1* to *host2*. It first requests a home base to perform an `update`(*fhns:drifters:NOMAD*, *host2*) operation and then NOMAD moves to the new host. The agent framework is required to provide a mechanism to handle messages in transit that were sent to the old location of the agent. One way of handling this is for the agent to leave behind a forwarding agent at the old location storing messages while the agent is migrating and redirecting them to the new location upon the agent's request. The forwarding agent could be deleted by expiration or from a request made by the originating agent.

When the agents are done performing their tasks, they are able to remove themselves from the system by issuing a `delete`(<agent absolute name>) request.

When a home base crashes and is removed from the network, the agents using that home base still need to have some way of being located. The agents are still on the network but the locating mechanism has been disrupted, and not addressing this contingency would render these agents *unreachable*. Names that are location-independent lend themselves nicely to fault tolerance since they are not bound to any home base explicitly. Using Chord, we start with a logical ring and place home bases at distinct points along the ring based upon their corresponding identifiers. Moving in a clockwise fashion, the subsequent home bases keep copies of the mapping records belonging to the previous home bases. So in the event that a home base fails,

the subsequent home base will be able to service all the requests that would have been handled by the now departed home base. Agents therefore do not become unreachable when their home base fails. The agents using location-dependent names can still be found because the host portion of their names corresponds to a home base and thus maps to an identifier. The subsequent identifier can easily be determined and the replicating home base can thus be found. This does introduce a incongruity in that the location-dependent name does not accurately give the location of the agent's home base, but there is no way to solve this problem since we wish names to be persistent even in the event of a failure. This incongruity can be avoided by using location-independent names.

3 IMPLEMENTATION

An instance of the Fault-Tolerant Home-Based Naming Service has been prototyped using Chord as the lookup service. The prototype was then analyzed using the World-Wide Computer (WWC) [20] as a testbed. What follows are brief descriptions of the WWC and Chord. The architecture of the prototype is then described.

3.1 Context: World-Wide Computer (WWC)

This framework recognizes the Internet's vast computing capability with its very name: the World-Wide Computer. The WWC enables tasks to be distributed among participating computers. The World-Wide Computer has been introduced to make it possible to pool the computing ability of devices that can communicate over a wide-area network such as the Internet. The WWC consists of a set of virtual machines, or *theaters*, hosting universal actors. The universal actors are reachable throughout the Internet by their globally unique Universal Actor Names (UAN). The UAN identifiers persist over the lifetime of an actor and are used to obtain the actor's current theater. The Universal Actor Locator (UAL) represents the location of this theater. Universal actors extend actors [1] in that they are able:

- to bind to a unique global name and to an initial theater,
- to obtain references to their peers from their names,
- to communicate through message passing in a location-independent manner, and
- to migrate from one theater to another.

In summation, the WWC is comprised of universal actors, theaters, and dedicated name servers.

An example of a UAN is

uan://wwc.publisher.com:3030/poemAgent.

The relative name of this actor is *poemAgent*, *wwc.publisher.com* is the name server or home base, and *uan* is the naming service protocol. The UAN has an identical format to the location-dependent names of FHNS. The naming server keeps track of where the agent is located by receiving information in the form of UALs. In this example, say the actor's UAL is

rmsp://wwc.editor.com:4040/poemAgent.

The naming server knows from this UAL that the poemAgent is located at the *wwc.editor.com* theater. The *rmsp* refers to a Remote Message Sending Protocol that enables sending messages to remote actors. The format for UANs has been augmented so as to support location-independent names for FHNS.

The default naming scheme for this framework uses a home-based strategy and location-dependent names. The name servers do not include any fault tolerance features. So when a name server fails, the agents that are bound to that name server become unreachable to collaborating agents holding only their names. FHNS is used to eliminate this single point of failure.

3.2 Context: Chord

Chord is a distributed lookup protocol that addresses the problem of efficiently locating data in peer-to-peer networks. The Chord protocol supports just one operation: given a key, it maps the key onto a node. The key can be associated with a particular data item and together both the key and data are stored at the node to which the key maps. The data is then located by a *lookup(key)* algorithm that yields the value associated with that key, which could be the IP address for a given node on the Internet.

The consistent hash function assigns each node and key (such as the name of an agent) in the system an m-bit identifier. The node identifiers can be established by hashing the IP address or domain name. Likewise the identifiers for the keys, which are strings, are established by hashing the keys themselves. As with any hash function, collisions can occur. Collisions between keys are unimportant since keys are allowed to map to the same identifier and it is the keys themselves that must be unique. Collisions between nodes must be avoided in order for Chord to work correctly. Chord does not prevent collisions.

Once the keys and nodes have been mapped to identifiers, the keys are then assigned to nodes. Any given key is stored at the first node whose identifier is equal to or greater to the key's identifier, where the last identifier in the identifier space is less-than the first identifier to effectively give a logical ring or an *identifier circle*.

Each node maintains a routing table with m entries, where m is the number of bits in the binary representation of the key or node identifiers. The routing table is called the node's *finger table* and the i^{th} entry in this table is given by $n+2^{i-1}$, where n is the identifier of the node and $1 \leq i \leq m$. The node storing the identifier given by the i^{th} entry is listed in the finger table as the node to forward requests to when the $n+2^{i-1}$ identifier is sought. Also listed in a node's finger table is the node previous to it in the identifier circle and is referred to as the *predecessor*.

If a node gets a lookup request for an identifier not listed in its finger table, it forwards the request to the entry in its finger table that lists an identifier closest to, but less than or equal to, the requested identifier.

In a dynamic network, nodes can join and leave at anytime. When a node joins the ring it has to request that an existing node provide the location of its nearest neighbors in the identifier space. Once the joining node receives its finger table information, it then requests that the keys it is responsible for be transferred. When a node leaves the network gracefully, it transfers its keys and has its neighbors update their finger tables accordingly. Nodes leave and join the Chord identifier circle at a cost, with high probability, of $O(log^2\ n)$ messages.

Each node's nearest neighbor or neighbors keeps copies of the keys belonging to that node. So when a node fails, the nearest neighbor of that node can service the *lookup* operation directed to the failed node. Chord does not specify the number of neighbors that should provide redundancy but this number does correspond to the amount of fault tolerance attainable. If ten neighbors provide redundancy, then the system can conceptually recover without losing any of the keys in the event of ten simultaneous node failures.

The nodes perform stabilizing procedures in which they determine if their predecessor and immediate successor information is correct. By performing stabilization, the immediate neighbors of a failed node will see that it is not responding and will take the appropriate measures to recover from the failure by updating their finger tables. Chord does not specify how often to perform stabilization; however, the Chord test-case implementation performed stabilization at an average rate of 0.5 invocations per minute.

3.3 FHNS Prototype Developed for the WWC

The WWC already has a naming service in place that uses location-dependent names. This naming service is first extended to use location-independent names. A UAN can take the traditional form, specifically

uan://<home base host>:<port>/<actor relative name>.

A UAN can also take the location-independent form,

uan:wwc:<actor relative name>.

The namespace identifier, for this prototype, is called *wwc* and the corresponding hashing function to be used will be *SHA-1* [21], which hashes the given inputs to a 160-bit identifier. The namespace will encompass all identifiers $0 <= i < 2^{160}$.

A unique type of theater has been developed to serve as the home bases for FHNS and are called *home base theaters*. The naming service has been incorporated into these theaters instead of using dedicated

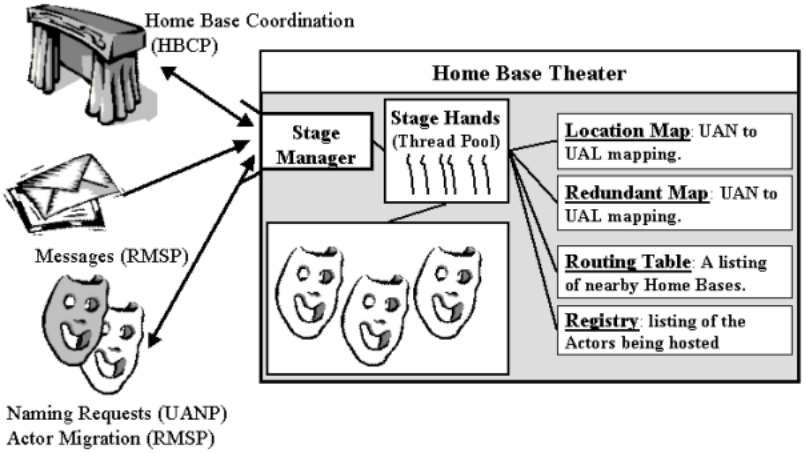


Figure 1: A Home Base Theater in the WWC.

naming servers. This is done to better ensure fault-tolerance. The alternative would be to have the actors keep a listing of home bases that they could query. A *Denial of Service* (DoS) could result if the home bases that an actor had a listing for had all failed. Having the actors direct all requests, with the exception of the `put` request, to the theater where they are presently "performing" ensures that the naming service is always available to the actors.

The test case consists of universal actors and home base theaters. These home base theaters host the universal actors and also serve as the name servers. Figure 1 depicts the structure of the home base theaters in the WWC. The figure also illustrates the other services provided by these theaters. In keeping with the "Broadway" manner of speaking, the home base theaters have a single *stage manager* that listens on a particular port for requests. Upon receiving a request, the stage manager delegates responsibility to servicing the request to a *stage hand* that is chosen from a thread pool. As illustrated in Figure 1 a home base theater can service a naming request using the Universal Actor Naming Protocol (UANP), a message directed to a hosted actor using the Remote Message Sending Protocol (RMSP), a hosting request also using RMSP, or a coordination request using the Home Base Coordination Protocol (HBCP).

The methods belonging to the Universal Actor Naming Protocol are identical to the methods in the API for the Fault-Tolerant Home-Based Naming Service. The Fault-Tolerant Home-Based Naming Service is thus capable of servicing all UANP requests. One modification to UANP that was needed was to change it from being a strictly text-based protocol to being able to use Java objects. This was done to facilitate communication in that all requests could come into the home base over the same port.

When a message is received via RMSP, the stage hand servicing the request obtains from the *registry* a reference to the universal actor to whom the message is intended and places the message in the actor's mailbox. A migrating actor comes into the home base theater through the same port as the other types of communications. The actor is registered with the home base and then is left to act upon the messages in its mailbox.

The Home Base Coordination Protocol provides the methods necessary for home bases to transfer needed information to participate in the namespace (logical ring). Table 3.3 details the different HBCP requests and corresponding services.

The system is initialized by a single home base theater being started. The domain name of the machine the home base theater is running on along with the port number is hashed according to the *wwc* namespace hashing algorithm and an identifier is assigned. Notice that the domain names are being hashed and not the IP addresses. This is not to say that the IP address cannot be used, they in fact can. However, for any given node either the domain name needs to be used in all cases or the IP address since the IP address and domain name will hash to different identifiers.

Other home base theaters can then join the system by contacting a home base theater that already belongs to the system and retrieving the necessary routing table information. A joining home base theater, or node

Request and Service	Description
NODEJOIN	Request made by a joining node where the servicing node returns with the requesting node's immediate successor
INITIALIZE	A joining node obtains its predecessor and routing entries from its successor
UPDATECOVERAGE	Joining or leaving node is inserted or removed and requests its successor and predecessor to update their Namespace coverage
GETKEYS	A joining node is provided the keys it is responsible for by its successor
REPLICATE	A node requests its successor to replicate the specified actor mapping
UPDATETABLE	A node has either joined or left (failed) and other nodes are informed to update their tables
GETNODE	Gets the node covering the region where the given Namespace identifier belongs
STABILIZE	A node periodically confirms the presence of its successor and predecessor

Table 4: Home Base Coordination Protocol

as it is referred in the proceeding explanations, first determines its identifier. It then issues a `nodejoin` request to an existing node. The servicing node returns with the name of the joining nodes immediate successor. The joining node then issues an `initialize` request to its immediate successor. The successor then provides the name of the joining node's predecessor and the nodes to list in its routing table and also the nodes that need to be informed of this node joining. The joining node then requests its successor provide the keys that it is responsible for with a `getkeys` request. Next the joining node issues `updatetable` requests to those nodes identified that need to be informed of the node's presence. Afterwards, the joining node tells its predecessor and immediate successor to update their coverage with an `updatecoverage` request. Upon completion of these requests, the joining node is now a part of the system and it begins its stabilization cycle. The stabilization cycle involves a node issuing a `stabilize` request to both its predecessor and successor. The predecessor returns with its predecessor and the successor returns with its successor. So in the event of a successor or predecessor failure, the node will already know of that particular node that will assume the role of the failed successor or predecessor. When the node, home base theater, receives a UANP `put`, `update`, or `delete` naming request, it sends a HBCP `replicate` request to its successor so as to keep the *location maps* and *redundant location maps* consistent.

Universal actors can enter the system at any point after the initial home base theater is up and listening for requests. The universal actors become reachable to other universal actors by issuing a `put`(*UAN*, *UAL*) request to one of the home base theaters.

4 PERFORMANCE ANALYSIS

The FHNS prototype has been put through different tests. The intent for these tests is to gather metrics that will support the requirements specified in Section 1 regarding FHNS being efficient, scalable, and robust. In the first test case, both location-independent and location-dependent names will be used. The results obtained from this first test case will pertain to resilience of FHNS in the event of a home base failure and is used to illustrate robustness. Messaging efficiency and scalability are evaluated in the next test case. The processing time to service requests for location-independent names versus location-dependent names is evaluated. Lastly, a scalability analysis pertaining to the load balancing capabilities is then detailed.

Before going further with the analysis, the reader should consider the specific implementation of the prototype. The tests that have been outlined are intended to measure the performance of the FHNS model; however, the tests predominately relate to the specific implementation of using Chord as the lookup service. Bearing the impact of Chord upon these results, the analysis of FHNS continues with the first test case regarding robustness.

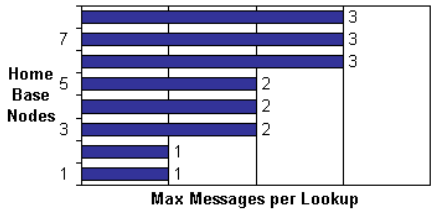


Figure 2: Single and Sequential Home Base Failures going from a total of 8 Home Bases to 1.

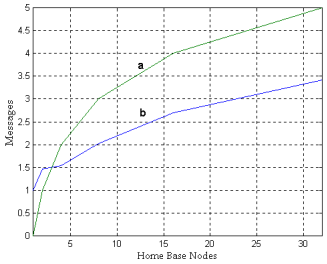


Figure 3: Message Count versus Number of Home Bases.

4.1 Robustness

The primary objective with using a lookup service such as Chord is to incorporate its fault tolerance features into FHNS. Since fault tolerance is the primary concern, this capability was tested to establish that the FHNS model is sound. We begin with 8 home bases and fail the home bases one by one, effectively degrading the distributed naming system to a centralized naming system. After each home base failure, `get` requests are made for all the actors in the system to confirm that they are all still reachable. The maximum number of messages it takes to have a request serviced is recorded. For this test, there are an equal number of actor location-dependent names as there are actors with location-independent names.

When the home base of an actor with a location-dependent name fails, FHNS hashes the domain name and port number for that failed home base and determines the immediate successor of the failed home base using a `getnode` request. Once the successor is determined, the request is then forwarded. It is known that the immediate successor of the failed home base has a redundant location map for the failed home base. The immediate successor is thus capable of servicing requests that were intended for its predecessor, which is the failed home base.

Figure 2 shows the number of messages it takes to find the back-up home base. The home bases refer to both their immediate location map and their redundant location map when servicing a request and this explains why only one message is needed when two home bases are present. The figure illustrates that FHNS can recover from all home bases failing except for one.

Performance regarding multiple home bases concurrently failing is not evaluated since it is known that this implementation has a single replication depth. Only a home base's immediate successor provides redundancy. If both the home base and its successor failed then there would be actors that would be rendered unreachable. It follows that if the replication depth was doubled, FHNS could recover from two consecutive and concurrent home bases failing. Increasing the replication depth, however, also has the disadvantage of increasing the number of HBCP `replication` requests.

The failed nodes are identified by the home bases periodically performing a stabilization routine. Stabilization involves a home base attempting to communicate with its predecessor and successor. If the home base is not able to communicate it takes the appropriate measures to replace the failed node and transfers and replicates the mapping records as needed.

4.2 Messaging Efficiency

Next we turn to the number of messages it takes for FHNS to service a request in a fault-free system. By messaging we count only the requests and forwarded requests and not the responses. When location-dependent names are used only a single message is needed. The messages needed for location-independent names depend on the lookup service employed. Stoica et al [16] state that with high probability, Chord resolves a lookup with $O(log\ n)$ messages. This assertion was confirmed by testing. Figure 3 shows the $log\ n$ trend with line a and line b is the average number of messages it took for FHNS, using Chord, to resolve a lookup pertaining to location-independent names. When only two home bases are present, it can take two messages to resolve a lookup; specifically the original request and the forwarded request. In the presence of n home bases, where $(n > 2)$, it takes on average less than $O(log\ n)$ messages.

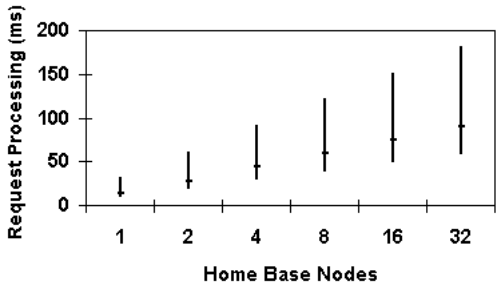


Figure 4: Prototype Request Processing Time when Using Location-Independent Names.

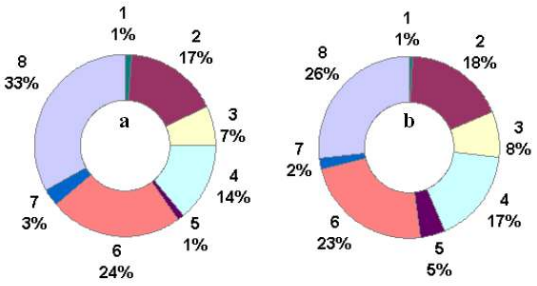


Figure 5: (a)The Mapping Records coverage and (b)Namespace coverage divided among 8 Home bases.

4.3 Time Efficiency

FHNS is indeed highly efficient when location-dependent names are used. Its efficiency is put into question with location-independent names. If we do not consider network communication, the *Request Processing Time* (RPT) when using location-independent names is over 3 times that of the processing for location-dependent names. FHNS takes on average 420μs to process a location-independent request and 130μs to process a location-dependent request. This time corresponds to the time elapsed from a home base receiving a request to the generation of a response.

If we consider communication over a local network, the total RPT is in the milliseconds range. It takes, on average, 14ms for FHNS to service a `put` request that requires no hops be taken. Approximately half this time is spent communicating with the requester and the other half communicating a `replicate` request. When hops are necessitated, the latency is increased by the number of hops. Figure 4 shows the minimum and maximum RPTs experienced by FHNS when servicing location-independent `put` requests. Due to these latencies, the location-dependent names are the preferred format for actor names, since they provide fault-tolerance without giving up on efficiency.

4.4 Load Balancing

To fully meet the efficiency and scalability requirements, the home bases can not be too overloaded, else there is the potential for a bottleneck. By using Chord as the lookup service, FHNS is able to attain load balancing. Ideally, load balancing entails that given K identifiers and H home bases, each home base is responsible for K/H identifiers[3]. In practice, however, the number of identifiers and mapping records per home base exhibits large variations. For instance, when the FHNS prototype consists of 2 home bases, one of the home bases covers 53% of the namespace identifiers and the other home base covers the remaining 47%. In contrast when two additional home bases are added giving a total of 4, 2 of these home bases each only cover 1% of the namespace identifiers, another covers 38%, and the last home base's coverage increases from 53% to 60%[4].

Figure 5 illustrates the namespace identifier coverage and the mapping record coverage. It can be seen from this figure that load balancing is not sufficiently achieved. It could be argued that load balancing is achieved by half the nodes but it certainly cannot be said for them all. Also depicted in the figure is the obvious correlation between namespace identifier coverage and mapping record coverage. The home bases with larger namespace coverage also store more of the mapping records.

The region for a node begins at the node's predecessor's identifier and spans all identifiers up to the node's identifier. Stocia et al [16] propose that each node (home base) divide the namespace into $(O(log\ n)\ +\ 1)$ regions of coverage. In order to do this, the nodes would each have $O(log\ n)$ virtual nodes. Applying this proposal has the effect of "balancing" the coverage amongst the nodes. Figure 6 depicts how using virtual nodes equalizes the coverage among the home bases. Although not necessarily ideal, a level of load balancing is thus achieved when using Chord as a lookup service and having each home base cover $(O(log\ n)\ +\ 1)$ regions. A node's total namespace coverage is the summation of the node's "non-virtual" coverage and its virtual coverage.

[3]Assumes all keys are equally "popular" and the nodes are homogeneous.
[4]The imbalance is due to two nodes producing very similar hashing values.

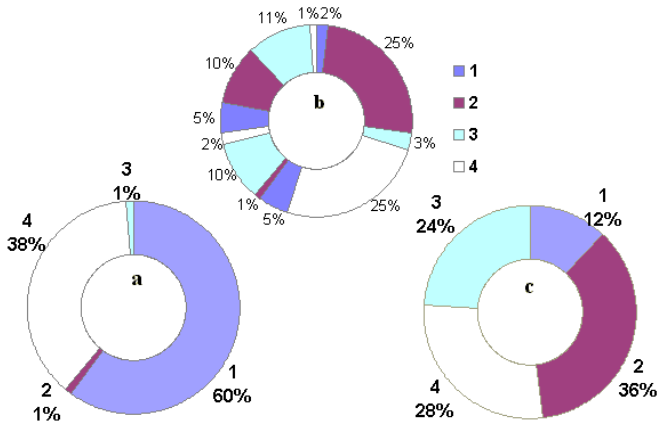


Figure 6: (a)Namespace coverage divided among 4 Home Bases, (b)Namespace Coverage divided among 4 Home Bases with 8 Virtual Nodes (c) Total Namespace Coverage with Virtual Nodes.

5 CONCLUSIONS

The Fault-Tolerant Home-Based Naming Service is a novel approach to locating mobile agents in a distributed computing framework. FHNS builds upon DNS and a peer-to-peer lookup service. And as a result of this meshing, this naming service is fault-tolerant as was shown by using a lookup service such as Chord. The Fault-Tolerant Home-Based Naming Service is also very efficient in resolving the location of a mobile agent.

The specific contributions made in this paper are as follows:

1. **A naming service specifically designed for use by mobile agents**. From the research that has been done, it was observed that efforts on the specific topic of naming services for mobile agents has not received much attention, at least when compared to mobile agent security, mobility, and messaging. Also from researching the naming services in current mobile agent platforms, it was observed that efforts do need to be focused on this very topic. Most Java-based agent systems use RMI or Corba which introduce single points of failure.

2. **A naming service that is highly efficient, fault tolerant, and scalable**. Location-dependent naming services, such as the one found in RMI for instance, are efficient in resolving a name to a location but introduce a single point of failure. FHNS is as efficient as these location-dependent naming services and it is also fault tolerant. Furthermore, the fault tolerance extension does not degrade the efficiency in resolving location-dependent names until a failure occurs.

3. **A naming service with more extensive fault tolerance than the conventional redundancy model**. Fault tolerance is a critical requirement of a naming service. Traditional redundancy schemes involve a designated primary server and an associated secondary server. The secondary server, however, does not necessarily have a back up. So when the primary fails, the system can no longer be described as being fault tolerant because the secondary, or the new primary, server can not fail. With FHNS, each server has a backup and each server is a backup for another server. The servers can fail down to a single remaining server and FHNS will still work as needed.

The FHNS implementation presented in this paper uses Chord, however other distributed hash table lookup services (e.g.CAN [13], Tapestry [22], Pastry [15], SPRR [9]) could be used to obtain different performance and reliability characteristics. [18] gives a detailed account of these alternatives.

There are numerous mobile agent systems currently in development (e.g, [6, 8, 12, 19]). Many of the systems are Java-based and have agents that communicate directly with other agents. Direct communication requires mobile agent naming and location mechanisms. The naming mechanisms currently employed by these mobile agent systems are not as efficient as they could be and/or they are not robust. FHNS could be incorporated by these mobile agent systems to realize an efficient and robust naming mechanism.

References

[1] G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, 1986.

[2] Tim Berners-Lee. Uniform Resource Locators (URL). Informational RFC 1738, 1994.

[3] Tim Berners-Lee. Uniform Resource Identifiers (URI): Generic Syntax. Informational RFC 2396, 1998.

[4] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building peer-to-peer systems with Chord, a distributed lookup service. In *Proc. 8 Wshop. Hot Topics in Operating Syst., (HOTOSVIII)*, pages 81–86, 2001.

[5] G. H. Forman and J. Zahorjan. The challenges of mobile computing. Technical Report TR-93-11-03, IEEE Computer, 1993.

[6] Mitsubishi Electric ITA. Concordia: An infrastructure for collaborating mobile agents.

[7] David Karger, Eric Lehman, Tom Leighton, Mathhew Levine, Daniel Lewin, and Rina Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symposium on Theory of Computing*, pages 654–663, May 1997.

[8] D.B. Lange and M. Oshima. *Programming and Deploying Mobile Agents with Java Aglets*. Addison Wesley Longman INC, 1998.

[9] Xiaozhou Li and C. Greg Plaxton. On name resolution in peer-to-peer networks. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 82–89. ACM Press, 2002.

[10] R. Moats. URN Syntax. Informational RFC 2141, 1997.

[11] P. Mockapetris. Domain Names - Implementation and Specification. RFC 1035, 1987.

[12] Tomasz Muldner and Thian Tin Ter. Building Infrastructure for Mobile Software Agents. *World Conference on the WWW and Internet WEBNETC*, 2000:419–424, 2000.

[13] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[14] V. Roth. Scalable and secure global name services for mobile agents. In *6th ECOOP Workshop on Mobile Object Systems: Operating System Support, Security and Programming Languages*, 2000.

[15] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329+, 2001.

[16] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.

[17] Karim Taha and Thomi Pilioura. Agent Naming and Locating: Impact On Agent Design. In D. Tsichritzis, editor, *Trusted objects. - Geneve : Centre Universitaire d'Informatique*, pages 149–169. 1999.

[18] Camron Tolman. A fault-tolerant home-based naming service for mobile agents. Master's thesis, Rensselaer Polytechnic Institute, 2003.

[19] Maarten van Steen, Franz J. Hauck, and Andrew S. Tanenbaum. A model for worldwide tracking of distributed objects. In *Proceedings of the TINA'96 Conference*, 1996.

[20] Carlos Varela. *Worldwide Computing with Universal Actors: Linguistic Abstractions for Naming, Migration, and Coordination*. PhD thesis, University of Illinois at Urbana-Champaign, 2001.

[21] Fips 180-1. secure hashing standard. U.S. Department of Commerce/NTIS, National Technical Information Services, April 1995.

[22] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.

Diseño de un Soporte Didáctico para Asistentes Personales Digitales que Incorporan la Tecnología Bluetooth

María E. Villapol
Universidad Central de Venezuela, Facultad de Ciencias, Escuela de Computación,
Av. Los Ilustres, Los Chaguaramos, Caracas, Venezuela
Tel: +212 58 605 1132, Fax: +212 58 605 1131
mvillap@strix.ciens.ucv.ve

Jossie Zambrano
Universidad Central de Venezuela, Facultad de Ciencias, Escuela de Computación,
Av. Los Ilustres, Los Chaguaramos, Caracas, Venezuela
Tel: +212 58 605 1132, Fax: +212 58 605 1131
jossie@strix.ciens.ucv.ve

y

Mayerling Marquez
Universidad Central de Venezuela, Facultad de Ciencias, Escuela de Computación,
Av. Los Ilustres, Los Chaguaramos, Caracas, Venezuela
Tel: +212 58 605 1132, Fax: +212 58 605 1131
mayer@cantv.net

Abstract

Personal Digital Assistants (PDAs) are small portable computers which have special components similar to pencils (*stylus*) which users can use to interact with them; they usually incorporate wireless communication facilities. One of these wireless communication facilities is Bluetooth, which provides communication between devices via radio frequency in an area of around the 10 meters. The objective of this paper is to present the design and implementation of a tool for didactic support developed for children in ages between 7 and 8 years and the results of its use. The tool also incorporates the wireless communication via Bluetooth, this is used for supporting the communication between the teacher and the students. To where we know this is the first tool for didactic support for PDAs that incorporates the use of Bluetooth developed in Venezuela. The results show that it is possible to integrate the PDAs to the education, shortening the breach between the new computer technologies and the education, offering to motivation and new experiences.

Keywords: Personal Digital Assistants (PDAs), Bluetooth, Didactic Support.

Resumen

Los *Asistentes Digitales Personales (PDAs)* son pequeños computadores portátiles con los que se interactúa a través de un dispositivo especial similar a un lápiz (*stylus*) y que usualmente cuentan con facilidades de comunicación en particular inalámbricas. Una de las facilidades de comunicación usadas en los PDAs es Bluetooth, la cual es una tecnología de comunicación que proporcionar comunicación entre dispositivos vía radio frecuencias en un área de alrededor de los 10 metros. El objetivo de este artículo es presentar el diseño e implementación de una herramienta de soporte didáctico desarrollada para niños en edades comprendidas entre 7 y 8 años y los resultados de su utilización. La herramienta incorpora igualmente la comunicación inalámbrica vía Bluetooth, esta se realiza entre el docente y los alumnos. Hasta donde sabemos esta es la primera herramienta de soporte didáctico para PDAs que incorpora el uso de Bluetooth desarrollada en Venezuela. Los resultados muestran que es posible integrar los PDAs a la educación, acortando la brecha con las nuevas tecnologías informáticas, brindando motivación y nuevas experiencias.

Palabras claves: Asistentes Personales Digitales, Bluetooth, Soporte Didáctico.

1 Introducción

Los *Asistentes Personales Digitales* (*Personal Digital Assistants, PDAs*) son pequeños computadores portátiles con los que se interactúa a través de un dispositivo especial similar a un lápiz. A diferencia de las agendas electrónicas clásicas, pueden instalárseles diversas aplicaciones, similarmente que a un computador convencional. Su poder computacional y capacidad de almacenamiento se han ido incrementando notablemente en los últimos años a medida que sus precios se han hecho bastantes competitivos y accesibles. Estos equipos también cuentan con facilidades de comunicación en particular inalámbricas. Una de sus características más atractiva es su portabilidad debido a su bajo peso y pequeño tamaño.

Bluetooth Wíreless Technology es una tecnología de comunicación orientada a las *Redes de Área Personal Inalámbricas* (*Wíreless Personal Area Networks, WPANs*) [7], caracterizadas por proporcionar comunicación entre dispositivos vía radio frecuencias en un área muy limitada de alrededor de los 10 metros. Bluetooth esta siendo cada día más incorporado en dispositivos móviles tales comos los PDAs y teléfonos celulares, gracias a su tamaño y costo.

Las características de los PDAs, tales como costo, poder de procesamiento, capacidad de almacenamiento, capacidad de procesamiento de información multimedia, entre otras, los hacen una herramienta que puede ser usada en la educación y en particular la educación primaria. Adicionalmente, Bluetooth proporciona una forma sencilla de comunicación entre dispositivos a corta distancia que ya viene integrada en muchos de los PDAs. Por lo tanto, esta tecnología incorporada en los PDAs puede usarse en proyectos educativos que involucren la comunicación entre los participantes. Así el objetivo de este artículo es presentar el diseño e implementación de una herramienta de soporte didáctico desarrollada para niños en edades comprendidas entre 7 y 8 años de educación primaria y los resultados de de la utilización de la misma por los niños y maestros. La herramienta incorpora igualmente la comunicación inalámbrica vía Bluetooth en cada una de las actividades incluidas.

Cabe destacar que existen diversos trabajos que buscan incorporar el uso de PDAs en la escuela. Por ejemplo, Gustavo Zurita [11] diseñó e implementó una herramienta basada en el Aprendizaje Colaborativo con tecnología Inalámbrica 802.11 para niños con edades comprendidas entre 6 y 7 años. Tatar et al. [8] presentan una serie de proyectos que buscan explorar el uso de PDAs en un salon de clase para alumnos del año k-12. Ninguno de estos trabajos está específicamente orientado al uso de la tecnología Bluetooth.

Con la finalidad de alcanzar los objetivos planteados, este articulo esta organizado de la siguiente manera. En la Sección II, se da una breve introducción a los PDAs; mientras que en la Sección III se introduce Bluetooth. En la Sección IV se describe la aplicación desarrollada, mientras que la en Sección V se describen los detalles de la implementación y pruebas. Los resultados de utilizar esta herramienta se presentan en la Sección VI. Finalmente, la Sección VII presenta las conclusiones, recomendaciones y trabajos futuros generados por este trabajo.

2 Asistentes Personales Digitales

Los PDAs, también llamados computadores de la palma de la mano, fueron diseñados originalmente como organizadores personales. Las características básicas de cualquier PDA son una agenda, libreta de direcciones, lista de tareas, y block de notas, aunque con el mejoramiento de la tecnología, en lo que respecta al poder de procesamiento, capacidad de memoria y conectividad, ha hecho que el número de aplicaciones y áreas donde estos dispositivos pueden usarse se haya extendido.

2.1 Componentes

Un PDA básicamente consta de un procesador o CPU, memoria, un chip de sonido, un circuito de entrada y salida de audio, y conectores para la sincronización. Adicionalmente, el dispositivo tiene una pantalla incorporada. Los PDAs también cuenta con puertos de entrada y salida, que se usan para realizar la sincronización vía cable con el computador, la conexión de tarjetas de expansión de memoria y la conexión de dispositivos tales como cámaras. Uno de las características principales de los PDAs es la Interfaz Principal de Entrada que funciona por medio del uso de un lápiz especial (*stylus*), que permite ingresar datos a la computadora. Adicionalmente, existe una amplia variedad de periféricos que pueden ser acoplados a las PDA, entre ellos se pueden mencionar: módems, cámaras fotográficas, cámaras de video, impresoras, almacenamiento externo y escáner.

Una característica importante de los PDAs es su portabilidad. Esta se facilita gracias a una batería interna o a las pilas, que se pueden cambiar por baterías recargables, que tienen estos equipos. En algunas PDAs, una pila botón asegura la conservación de los datos en la memoria RAM cuando la alimentación principal se agota.

2.2 Sincronización

Otra de las características relevantes de los PDAs es el hecho de poder sincronizar la información y aplicaciones que existen en el dispositivo con las que existen en el computador. Esto permite tener una copia de los mismos en ambos dispositivos. La forma de realizar esto es a través de una base o cable que se puede conectar a algunos de los puertos estándares del computador (puerto serial, puerto USB). La sincronización también puede llevarse a cabo vía inalámbrica usando tecnologías tales como Bluetooth o IrDA. Adicionalmente debe existir un software encargado de manejar este proceso de sincronización.

2.3 Comunicación Inalámbrica

Debido a que uno de los objetivos principales de los PDAs es permitir la comunicación móvil, estos dispositivos vienen con facilidades de comunicación inalámbricas. Los mismos pueden incorporan facilidades para la comunicación a corta distancia, a distancias locales o a largas distancias. Entre las facilidades para la comunicación a corta distancia se encuentran Bluetooth e IrDA [7], mientras que en el ámbito local se encuentran las tecnologías basadas en el estándar 802.11 [7]. Para la comunicación a larga distancia, algunos PDAs vienen con capacidades de telefonía celular o se pueden comunicar usando un dispositivo denominado MODEM a través de la red de telefonía celular.

2.4 Sistema Operativo

Existen varios sistemas operativos para PDAs. Ellos están muy ligados a la marca y modelo de este dispositivo. Entre los mas usados se encuentra el Palm OS que se encuentra en los computadores comercializados por Palm y Handspring [6]. El otro es el Windows Mobile desarrollado por Microsoft. Adicionalmente, existe una versión de Linux para PDAs.

3 Bluetooth Wireless Technology

Bluetooth Wireless Technology es una tecnología de radio frecuencia (*Radio Fraquency, RF*) que ofrece conectividad a corta distancia para equipos personales, portables, PDAs, entre otros. Bluetooth esta orientado al reemplazo de interfaces tradicionales, tales como RS-232 y conectores propietarios, proporciona una interfaz uniforme para acceder servicios de voz y datos, proporciona acceso a una red de área amplia usando un gateway personal, tal como un teléfono celular, y proporciona una comunicación sin infraestructura, que se puede usar para el soporte a grupos colaborativos (reuniones, conferencias) [1].

3.1 Pila de Protocolos

La especificación de Bluetooth incluye una especificación del núcleo que describe los detalles de los diversos protocolos que conforman la pila de protocolos; y una especificación de perfiles que incluye los detalles del uso de la tecnología para soportar varias aplicaciones e indica cuales de los aspectos de la especificación del núcleo son obligatorios, opcionales y no aplicables. La Figura 1 muestra la pila de protocolos que conforman el estándar. La misma divide los protocolos en los siguientes niveles:

a) **Protocolos fundamentales de Bluetooth (protocolos del núcleo):** son específicos de Bluetooth y han sido desarrollados por el *Grupo de Interés Especial* (*Special Interest Group, SIG)* de Bluetooth.
b) **Protocolos de sustitución de cable:** suministran señalización de control que emulan el tipo de señalización que se asocia usualmente con los enlaces de cable
c) **Protocolos de control de telefonía:** definen la señalización de control de llamada para establecer llamadas de voz y datos con dispositivos Bluetooth. También define un protocolo (Comandos AT) que definen como pueden controlarse un MODEM y un teléfono móvil.
d) **Protocolos adoptados:** son protocolos existentes que se utilizan para diversos fines en las capas superiores.

3.2 Protocolos Fundamentales

La especificación de Bluetooth establece el uso de la frecuencia de 2.4 GHz (más específicamente la banda de frecuencia en la mayoría de países es de 2.4 – 2.4835 GHz). Definiéndose 79 canales físicos de 1 MHz sobre esta banda, siendo la tasa de transmisión es de 1 Mbps.

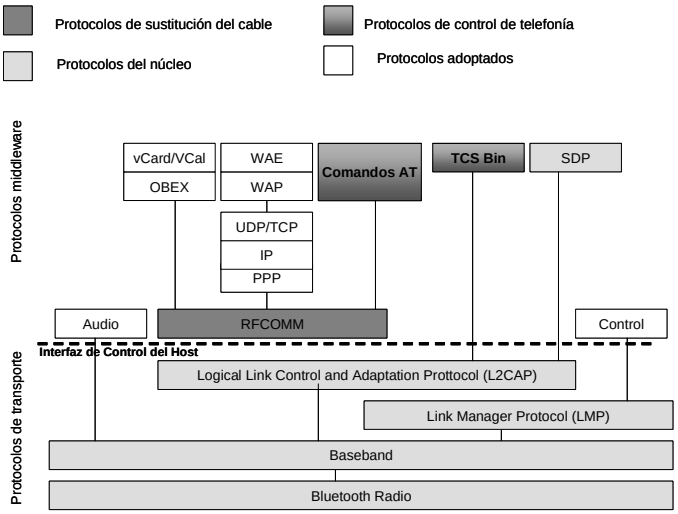


Figura 1: Pila de Protocolos de Bluetooth.

En Bluetooth, una *piconet* es una colección de dispositivos que pueden comunicarse. La *piconet* se forma de una forma ad hoc y contiene un dispositivo maestro y a lo sumo 7 dispositivos esclavos. Adicionalmente, un dispositivo en una *piconet* puede ser parte de otra *piconet* (como maestro o esclavo). Esta especie de solapamiento se conoce como *scatternet.*

La transmisión de la señal ocurre usando una técnica de saltos de frecuencia elegidos de forma aleatoria, entre los 79 canales físicos de 1 MHz. Ya que la tasa de salto es de 1600 saltos / seg cada canal es ocupado por 0,625 ms. Este período es llamado *slot*, los cuales están numerados secuencialmente. Adicionalmente, los dispositivos (radios) se comunican usando *Time Division Duplex (TDD)* [7], donde la data es transmitida en una dirección a la vez y la transmisión se alterna entre las dos direcciones. La frecuencia de salto es determinada por el maestro.

Existen dos tipos de enlaces que se pueden establecer entre el maestro y un esclavo: uno *Sincrono Orientado a Conexión (SCO)* para el tráfico con restricciones de tiempo (audio), donde se reserva un ancho de banda fijo en una conexión punto-a-punto (M/E) a intervalos regulares de tiempo. El maestro reserva *slots* (en pares, uno por cada dirección). El otro es el *Asíncrono no Orientado a Conexión (ACL)* y esta destinado a tráfico de mejor esfuerzo sin ninguna restricción de tiempo. La comunicación se realiza en *slots* no reservados para el tráfico SCO.

La operación de Bluetooth se basa en el establecimiento de una conexión, gestión de la conexión y desconexión. Durante el procedimiento de establecimiento de la conexión no se han establecidos los roles del maestro ni esclavo. Durante este periodo los dispositivos entran en un proceso de indagación para ver que dispositivos están en el rango. Una vez que se han identificado los dispositivos, el dispositivo puede entrar en un procedimiento de *page*, tendiente a establecer la frecuencia de salto y así pasar al estado de conectado. Durante la conexión un dispositivo puede estar en varios estados: un estado activo donde participa en una *piconet*. En este estado escucha, transmite y recibe paquetes; un estado de husmeo (*Sniff*) donde escucha en *slots* específicos; un estado de sostenido (*hold*) que es un estado de potencia reducida, donde aun puede participar en el intercambio de paquetes SCO; y un estado de estacionado (*park*) donde no participa en la *piconet*, pero es retenido como parte de ella. Finalmente, el dispositivo puede desconectarse en cualquier momento.

Bluetooth establece ciertos mecanismos de seguridad [2][4]. Uno es la *autenticación* que tiene como finalidad verificar identidades de las unidades involucradas en el procedimiento. Y el otro es el *cifrado* donde la información del usuario puede ser protegida cifrando el paquete; sin embargo, el código de acceso y el encabezado del paquete nunca se cifran.

Adicionalmente a las capas anteriores, existen dos protocolos más que conforman los protocolos fundamentales de Bluetooth. El primero es el *Manejador del Enlace (Link Manager Protocol, LMP)*, que es el encargado de gestionar diversos aspectos del enlace de radio entre el maestro y el esclavo. Se entiende por enlace la conexión física establecida entre los dispositivos. El otro es el L2CAP, el cual es un protocolo de la capa de enlace entre entidades con un número de servicios. Esta capa confía en los protocolos de las capas más bajas para el control de error y flujo y hace uso de los enlaces ACL pero no soporta enlaces SCO.

3.3 Perfiles de Bluetooth

Los perfiles de Bluetooth definen los protocolos y las características que soporta un *modelo de uso*. Un modelo de uso es un conjunto de protocolos que implementan una aplicación particular basada en Bluetooth. Los perfiles se pueden dividir en [4]: generales y específicos. Entre los perfiles generales se encuentra el perfil de acceso genérico que define los procedimientos genéricos para descubrir dispositivos Bluetooth y aspectos relacionados a la gestión de enlace para dispositivos que están estableciendo una conexión. El perfil del puerto en serial establece como deben configurarse los dispositivos Bluetooth para emular una conexión serial usando RFCOMM. RFCOMM permite que aplicaciones que han sido diseñadas e implementadas para operar sobre cables seriales corran sobre Bluetooth sin ser modificadas. El perfil de aplicación de descubrimiento de servicios describe como un dispositivo puede descubrir servicios registrados en otros dispositivos Bluetooth al igual que otra información acerca de estos servicios. El perfil genérico de intercambio de objetos define como los objetos pueden ser intercambiados usando el protocolo OBEX [7].

Varios perfiles específicos han sido definidos [4], entre ellos se encuentran el perfil de transferencia de archivos que ofrece la capacidad de transferir objetos de datos de un dispositivo a otro tales como hoja de cálculo, presentaciones, imágenes. El perfil de acceso a una red de área local le permite a un dispositivo Bluetooth acceder a una LAN, tal como si estuviera conectado a la red. El perfil de sincronización define los requerimientos de la aplicación para los dispositivos Bluetooth para el soporte del modelo de uso de sincronización. Otros perfiles se encuentran descritos en [2][4].

4 Análisis y Diseño de la Herramienta

La herramienta de soporte didáctico fue desarrollada en base al Currículo Básico Nacional Venezolano [3], el cual comprende los aspectos y consideraciones concernientes al pensum de estudio de la educación básica (de primero a sexto grado) en Venezuela. Se definieron actividades dentro de este programa que fuesen adaptables al objetivo general planteado en este trabajo con la ayuda del personal docente del segundo grado de Educación Básica del Colegio La Consolación de Santa Mónica, ubicado en Caracas. En esta etapa fue necesario instruir y motivar al personal del área educativa acerca de las características del dispositivo, haciendo énfasis en su capacidad de comunicación inalámbrica.

Para la selección de las actividades educativas se llevaron a cabo reuniones donde se interactuó con los coordinadores y personal docente de la institución, donde se observaron situaciones educativas donde era posible aportar nuevas experiencias a los alumnos a través de un soporte didáctico. Las actividades elegidas se encuentran enmarcadas en las siguientes áreas del saber: matemática, comprensión lectora e inglés [10]. Estas áreas se escogieron con el propósito de crear un soporte didáctico variado y funcional que permitiese a los alumnos practicar a través de ejercicios.

Dentro de cada área que se definieron las siguientes actividades que se estaban ejecutando en el centro educativo al momento de desarrollar este trabajo [3]:

a) **Comprensión Lectora:** basado en el cuento de navidad “El Viejo Pascuero”.
b) **Matemática:** se basó en los ejercicios de la tabla de multiplicación del 2 y 3 y el valor de posiciones (unidad, decena, centena y unidad de mil).
c) **Inglés:** basado en el vocabulario básico de inglés correspondiente al nivel. El mismo se relaciona con: juguetes, frutas, partes del cuerpo, familia, animales. Para este trabajo se escogió el vocabulario relacionado con animales.

A continuación se describirá el diseño de la herramienta basado las siguientes fases:

a) **Fase I - Diseño del contenido:** se refiere al contenido de las actividades que se presentan en el soporte didáctico basado en las actividades cotidianas dentro del aula de clase. El contenido fue sugerido y evaluado por el personal docente del colegio.
b) **Fase II - Diseño de la interfaz:** para el diseño de la interfaz se tomó en cuenta cómo iba a ser la interacción entre el alumno y las actividades del soporte didáctico definidas en la etapa del análisis, así como también las características particulares del dispositivo utilizado. En este caso particular donde la pantalla del dispositivo es de tamaño 320x320 píxeles fue importante la ubicación y distribución de los elementos que componen la interfaz, puesto que un diseño adecuado de la misma contribuye a la motivación, fácil interacción, eficiencia y comprensión del soporte didáctico.
c) **Fase III - Diseño comunicacional:** se detalla la comunicación a través de mensajes informativos o de corrección para el alumno.

El soporte didáctico se dividió en los siguientes módulos: Aula Maestra, Aula Alumno, Viejo Pascuero, English Book, Multiplica y Cartel de Valores. A continuación se presenta cada módulo con sus tres fases de diseño.

4.1 Modulo 1 - Aula Maestra

4.1.1 Fase I Diseño de Contenido

Este modulo es manejado por el docente. A través de este se pueden activar los otros módulos que se describirán posteriormente. El también recibe la puntuación de los alumnos y puede, con imágenes interactuar, gracias a la comunicación inalámbrica (vía Bluetooth) para estimular a los alumnos por el resultado de las actividades. Además tiene una funcionalidad particular, que es la de consulta, que le permite al docente conocer los resultados de los módulos de cada alumno.

4.1.2 Fase II Diseño de la Interfaz

Una de las interfaces de este modulo se muestra en la Figura 2. Las demás interfaces no se muestran por limitaciones en el espacio disponible para este artículo. Como se dijo anteriormente a través de esta interfaz se activan los módulos de los alumnos vía Bluetooth, además de realizar consultas (*Consultar*) del puntaje obtenido por los alumnos y el otorgar premios (*Premiar*) a los alumnos.

Figura 2: Interfaz del menú en Aula Maestra.

4.1.3 Fase III Diseño Comunicacional

En la Figura 3 se muestran los mensajes comunicacionales de este modulo. *Info* muestra información sobre la comunicación inalámbrica. *Aula Alumno* es el mensaje que se recibe con la puntuación del alumno.

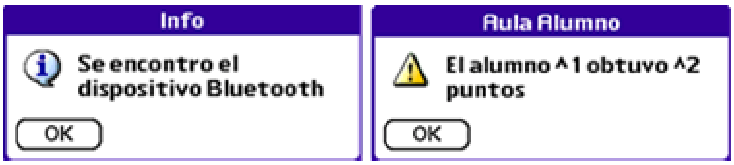


Figura 3: Mensajes Aula Maestra.

4.2 Modulo 2 - Aula Alumno

4.2.1 Fase I Diseño de Contenido

El alumno interactúa con la aplicación a través de este modulo. Este envía, a través de Bluetooth, información con respecto a la puntuación adquirida por el alumno al finalizar la actividad. Adicionalmente, él interactúa con Aula Maestra para activar algunos de los módulos correspondientes a las actividades de los alumnos.

4.2.2 Fase II Diseño de la Interfaz

En la Figura 4 se puede observar el diseño de la interfaz de este modulo. Inicialmente el programa le da la bienvenida (*Bienvenido*) al alumno y tiene una caja de texto para que el mismo ingrese su nombre. Entonces el alumno espera (*Esperando*) a que el docente le asigne el módulo de actividades que va a realizar.

Figura 4: Interfaz del Aula Alumno.

4.2.3 Fase III Diseño Comunicacional

Corresponde a la premiación que otorga el docente al alumno, luego que ha culminado la actividad y se ha recibido su puntuación como se muestra en la Figura 5. *Oro* le indica al alumno que realizó las actividades excelentes, *Plata* que realizó las actividades muy bien y *Bronce* que realizó las actividades bien.

Figura 5: Mensajes de Aula Alumno indicando la premiación obtenida por el alumno.

4.3 Modulo 3 - Una Carta al Viejo Pascuero

4.3.1 Fase I Diseño de Contenido

Este modulo incluye un cuento que trata de El Viejo Pascuero (San Nicolás) [9] y un conjunto de actividades destinadas a ejercitar la comprensión lectora de los alumnos.

4.3.2 Fase II Diseño de la Interfaz

En la Figura 6 se puede observar el diseño de interfaz de este modulo. Esta formada por una introducción gráfica (*Portada*) al modulo. Luego se muestra el contenido del cuento (*Texto*). A continuación siguen las actividades (*Actividad*) del modulo. Los botones de selección pueden ser texto o imágenes. Por motivos de espacio no se muestran todas las actividades.

Figura 6: Interfaz del Viejo Pascuero.

4.3.3 Fase III Diseño Comunicacional

En la Figura 7 se puede observar el contenido de los mensajes de este modulo. El mensaje de *Felicitaciones* se muestra cuando la respuesta es acertada, mientras que *Lo siento* se muestra cuando la respuesta no lo es.

Figura 7: Mensajes de Viejo Pascuero.

4.4 Modulo 4 - English Book

4.4.1 Fase I Diseño de Contenido

En esta actividad se presenta la imagen, el sonido y el nombre escrito de 17 animales en inglés como actividad complementaria de vocabulario dentro de la ésta asignatura.

4.4.2 Fase II Diseño de la Interfaz

En la Figura 8 se puede observar el diseño de interfaz de este modulo. *Portada* es la introducción gráfica al modulo. *Vocabulario* muestra la figura, la palabra escrita y una corneta como metáfora para reproducir el sonido. Se presenta una flecha hacia la derecha que permite ir a la siguiente pantalla del vocabulario, y una flecha hacia la izquierda que permite regresar a la pantalla anterior. En *Actividad* se pide seleccionar el nombre correcto del animal mostrado en la figura.

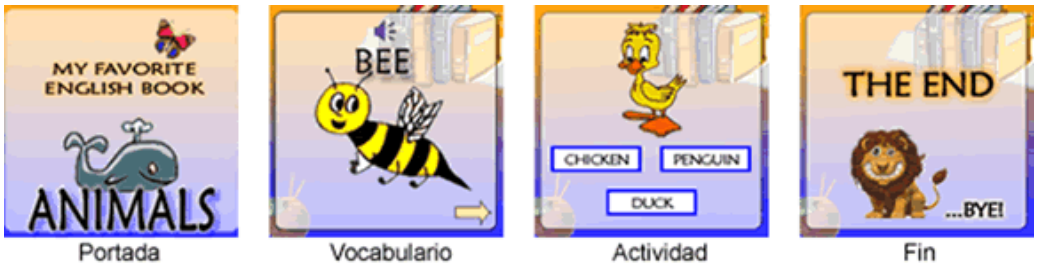


Figura 8: Interfaz de English Book.

4.4.3 Fase III Diseño Comunicacional

En la Figura 9 se pueden observar los mensajes de *Congratulations*, que se muestra cuando la respuesta es acertada, y *Try again* que se muestra cuando la respuesta no lo es.

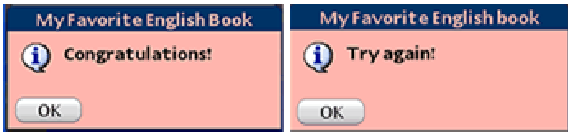


Figura 9: Mensajes de English Book.

4.5 Modulo 5- Multiplica

4.5.1 Fase I Diseño de Contenido

En esta actividad se hace una pequeña introducción del concepto de multiplicar (ver [10]), y dos ejemplos gráficos, para posteriormente plantear 10 ejercicios de las tablas del 2 y del 3 con un máximo de dos intentos.

4.5.2 Fase II Diseño de la Interfaz

En la Figura 10 se pueden observar algunas interfaces de este modulo. Inicialmente se presenta una introducción gráfica (*Portada*), luego la definición de multiplicar (*Definición*), y posteriormente un ejemplo gráfico de multiplicación (*Ejemplo*) y un ejercicio (*Ejercicio*).

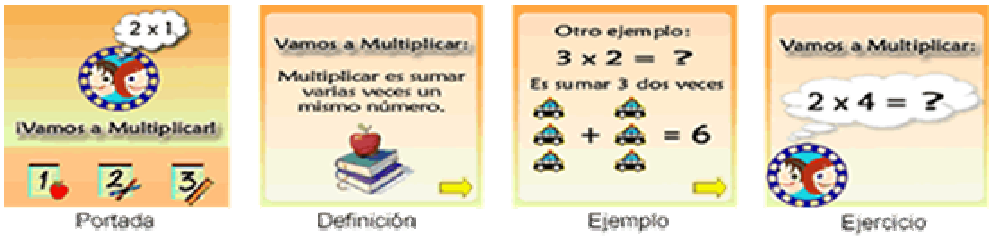


Figura 10: Interfaz de Multiplica.

4.5.3 Fase III Diseño Comunicacional

En la Figura 11 se pueden observar los mensajes de *Muy bien* que se muestra cuando la respuesta es acertada y *Intenta de nuevo* que se muestra cuando la respuesta no lo es.

Figura 11: Mensajes de Multiplica.

4.6 Modulo 6 - Cartel de Valores

4.6.1 Fase I Diseño de Contenido

En esta actividad se hace una pequeña introducción del concepto de valor de posición [3], y se plantean 10 ejercicios relacionados.

4.6.2 Fase II Diseño de la Interfaz

En la Figura 12 se puede observar el diseño de interfaz de este modulo, donde se muestra una introducción gráfica (*Portada*), la introducción sobre el cartel de valores (*Definición*) y el ejercicio (*Ejercicio*).

Figura 12: Interfaz de Cartel de Valores.

4.6.3 Fase III Diseño Comunicacional

En la Figura 13 se pueden observar los mensajes de *Muy bien* que se muestra cuando la respuesta es acertada y *Conocer la respuesta* que se muestra cuando la respuesta no lo es.

Figura 13: Mensajes de Cartel de Valores.

5 Implementación y Pruebas

A continuación se describen las actividades realizadas durante la implementación y pruebas de la herramienta:

a) **Implementación de los módulos:** Para la implementación del soporte didáctico se utilizó la herramienta Palm OS Developer Suite [5], que es un entorno de desarrollo para PC que permite realizar aplicaciones para dispositivos móviles con sistema operativo Palm OS. Adicionalmente, la aplicación fue desarrollada bajo el sistema operativo Palm OS versión 5.
b) **Revisión:** A medida que los módulos se fueron desarrollando fueron presentándose a los docentes, quienes dieron su apreciación en cuanto al contenido, disposición de los elementos en la interfaz, ortografía, redacción, comprensión de los mensajes y de las actividades.
c) **Introducción a la herramienta:** En esta fase se realizó una charla de aproximadamente 15 minutos de duración, cuya finalidad fue introducir a los alumnos los conceptos de dispositivos PDAs y tecnologías inalámbricas. Esta charla estuvo acompañada de un intercambio de ideas con los niños para captar sus opiniones e impresiones acerca de lo que vieron en la charla.
d) **Pruebas:** Las pruebas fueron realizadas en varias sesiones. La actividad se hizo en un salón fuera del aula de clases ya que los alumnos debían pasar en grupos de tres para realizar las actividades de los cuatro módulos (El Viejo Pascuero, English Book, Multiplica y Cartel de Valores). En estas pruebas participaron un docente, un observador, dos tecnólogos, y veintidós alumnos. Estas pruebas fueron complementadas con actividades de discusión, dibujos de la experiencia, entrevistas a los alumnos para que pudiesen exponer las ideas expresadas en las ilustraciones y al personal docente para registrar sus impresiones. En dichas pruebas se utilizaron 3 dispositivos de la marca Palm (dos modelo Tungsten T3 y uno modelo Tungsten T2 [6]).

6 Análisis de los Resultados

Los resultados obtenidos en las pruebas se analizaron desde los siguientes puntos de vista: *ámbito educativo*, donde se analiza el impacto de la tecnología y la herramienta en los alumnos y docentes; *soporte didáctico*, donde se analizan ciertos aspectos relacionados con la funcionalidad y la apariencia del soporte didáctico; *tecnología de comunicación inalámbrica* donde se analizan aspectos relacionados al uso de la tecnología inalámbrica en este tipo de actividad educativa; y finalmente *uso del PDA* que busca conocer el impacto de usar esta tecnología en la educación básica. Para cada uno de los aspectos antes mencionados se definieron una o más variables. De la Tabla 1 a la Tabla 4 se muestran las variables definidas, su definición y los resultados obtenidos.

Variable	Definición	Resultado
Navegabilidad	Se refiere a si el soporte contiene la secuencia apropiada en las actividades, orientando al alumno mientras este lo utiliza.	Se consideró que la navegabilidad del soporte didáctico fue adecuada.
Interfaz	Disposición de los elementos en la pantalla del PDA, colores utilizados, apariencia de mensajes, actividades y estímulos tanto del rol docente como del rol alumno.	Los docentes manifestaron que la interfaz les resultó amigable y adecuada para la edad de los alumnos tanto en la apariencia como en el lenguaje utilizado. Con respecto a la interfaz que manejaba el docente, expresaron que les pareció agradable y sencilla.
Usabilidad	Facilidad de uso del soporte didáctico.	Se concluyó que de manera general el soporte didáctico fue fácil de usar para los niños. Los docentes expresaron que les resultó sencillo e intuitivo el manejo del soporte.
Funcionalidad	Se refiere a si cada uno de los elementos incluidos en el soporte (tal como botones) cumple con la función que tiene definida.	Se determinó que el soporte didáctico fue funcional, ya que los elementos cumplieron con el objetivo para los que fueron incluidos.

Tabla 1: Resultados del análisis del soporte didáctico.

Variable	Definición	Resultado
Relacionadas con el Alumno		
Motricidad fina	La capacidad viso–motora del niño, vinculada a la coordinación y precisión en la utilización del *Stylus* de las Palm durante la realización de la actividad.	De manera general se observó que los niños que realizaron las pruebas tenían desarrolladas las destrezas viso-motrices necesarias para realizar las actividades satisfactoriamente.
Tiempo	La cantidad de tiempo que le toma al alumno concluir las actividades.	Se observó que el tiempo de culminación de la actividad fue muy variable. Esto dependió de distintas circunstancias como instrucciones y estabilidad de los equipos.
Comprensión	Proceso cognitivo que lleva al alumno a entender tanto las instrucciones como los ejercicios del soporte didáctico.	Se observó que los alumnos en su totalidad culminaron las actividades, lo cual lleva a concluir que hubo comprensión del soporte didáctico.
Atención	La concentración del niño durante la realización de las actividades.	Los niños en forma general prestaron atención tanto a la actividad como durante la charla introductoria.
Motivación	Interés que manifiesta el niño en realizar la actividad.	Tomando en cuenta la atención de los niños, y los dibujos que realizaron después de la experiencia; se concluyó que hubo una gran motivación y emoción hacia la actividad
Relacionadas con el Docente		
Puntuación	Puntaje obtenido por los alumnos en cada actividad (en una escala del 1 al 20).	La puntuación registrada fue útil para el docente, puesto que le permitió analizar de una manera cuantificable los conocimientos de los niños. Esto hizo posible hacer un sondeo del desenvolvimiento general del grupo en cada uno de los temas presentes en el soporte didáctico.
Motivación	Interés y disposición del docente en participar en la actividad.	El docente se mostró motivado y dispuesto a colaborar. En particular hizo notar que se sintió cómodo con la utilización del soporte didáctico ya que consideró que formaba parte importante dentro de la dinámica del mismo, a diferencia de otras actividades de computación que buscan reemplazar el rol del docente

Tabla 2: Resultados del análisis en el ámbito educativo.

Variable	Definición	Resultado
Alcance	Área de cobertura de Bluetooth en la actividad que se desarrollo entre el docente y los alumnos.	El soporte didáctico se utilizó dentro de un salón cuya extensión correspondía con el área recomendada de 10mts. Se concluyó que era una distancia apropiada para realizar las actividades dentro de un salón de clase.
Estabilidad	Confiabilidad del enlace durante la comunicación de los dispositivos.	En cuanto a la conexión Bluetooth no se obtuvo la estabilidad esperada, ya que durante las pruebas se presentaron fallas en la comunicación y reconocimiento de los dispositivos. Esto se tradujo en retrasos en la continuidad de las pruebas, incertidumbre y fatiga tanto en los niños como en los docentes mientras se solucionaban los inconvenientes técnicos.
Capacidad	Tasa de transferencia de datos y tamaño de los datos.	La transferencia de los datos durante la realización de las pruebas fue satisfactoria, solo viéndose afectada eventualmente por las fallas descritas en el punto anterior. La transferencia del puntaje y la premiación de las actividades, se dio de forma rápida y efectiva.
Compatibilidad	Grado de compatibilidad de los dispositivos PDA con el protocolo de comunicación inalámbrica Bluetooth.	Se determinó que los modelos de PDA utilizados fueron compatibles con la utilización de Bluetooth para la comunicación entre ellos.

Tabla 3: Resultados del análisis del uso de la tecnología inalámbrica.

Variable	Definición	Resultado
Manejo	Facilidad para operar el PDA.	Los alumnos y el docente manejaron rápida y fácilmente el dispositivo.

Tabla 4: Resultados del análisis del uso del PDA.

7 Conclusiones

Este trabajo presenta el diseño e implementación de una herramienta de soporte didáctico para PDAs, la cual utiliza la tecnología inalámbrica Bluetooth. La misma sirve como herramienta de ejercitación y evaluación de actividades escolares pertenecientes al segundo grado de educación básica en Venezuela. El contenido de esta herramienta pertenece a tres áreas distintas del conocimiento: comprensión lectora, vocabulario de inglés y matemática.

Las principales características de la herramienta son: le permite al docente conocer los resultados de forma cuantitativa, apreciar el desenvolvimiento del alumno durante la evaluación en un corto tiempo y obtener una visión detallada del grupo. Para los alumnos, la misma resultó motivadora ya que además de ser evaluados se involucraron con nuevas tecnologías informáticas, apoyándose en los conocimientos adquiridos previamente. La actividad fue definida por ellos como "una forma divertida de estudiar".

Los resultados del uso de la herramienta muestran que ni los niños ni los alumnos presentaron mayor inconveniente en el uso de la misma y del dispositivo, pudiendo culminar las actividades diseñadas sin mayores inconvenientes. Adicionalmente, la misma resultó altamente motivadora tanto para el docente como para el alumno. En cuanto al soporte didáctico, el mismo fue adecuado desde el punto de vista funcional, de interfaz, navegabilidad y usabilidad. En lo que respecta a la tecnología inalámbrica, se observó su importante rol para la participación tanto del docente como del niño en el conjunto de actividades planteadas. La cobertura proporcionada por Bluetooth es adecuada al salón de clases. Sin embargo, debido a algunos problemas encontrados durante la comunicación entre equipos, se recomienda realizar varias pruebas de conectividad entre los mismos en el área destinada para realizar la actividad. Finalmente, ni los alumnos ni el docente tuvieron problemas con el manejo de los PDAs.

Las mayores limitaciones encontradas durante el desarrollo de este trabajo fueron primeramente, el desconocimiento del uso y de las capacidades de los PDAs por parte del sector educativo, lo cual se vio reflejado en desinterés y poca credibilidad por parte de la mayoría de los docentes que fueron consultados, hacia el posible aporte de éstos dispositivos en la educación. Esto trajo como consecuencia la necesidad de invertir gran cantidad de tiempo para hallar una escuela que brindará su apoyo en cuanto al contenido educativo del soporte didáctico y que además permitiese realizar las pruebas necesarias con los alumnos. Otra limitante, es que no se contó con la cantidad de equipos móviles suficientes para efectuar las pruebas con todo el salón de clase. Debido a esto fue necesario dividir el salón en grupos de dos alumnos para realizar la actividad, lo cual hizo que la actividad se extendiera más de lo estipulado para cada actividad.

Entre los trabajos futuros que podrían realizarse a partir de este trabajo se encuentra el estudio de la incorporación de otras tecnologías inalámbricas como 802.11[7] en actividades educativas, desarrollando otras aplicaciones que aprovechen sus características. También es recomendable hacer pruebas con otras poblaciones de usuarios, específicamente de grados superiores, ya que si bien quedó demostrado que a este nivel los niños son aptos para el uso de los dispositivos, se puede inferir que un mayor grado de madurez permitiría desarrollar otro tipo de actividades y usos en la educación los cuales sería importante investigar.

Referencias

[1] Bisdikian C. *An Overview of the Bluetooth Wireless Technology*. IEEE Communications Magazine. December 2002. pp 86-95.

[2] Bluetooth SIG, Inc. *Specification of the Bluetooth System version 2.0*. November 2003.

[3] Ministerio de Educación. Currículo Básico Nacional de la República Bolivariana de Venezuela.

[4] Muller N. *Tecnología Bluetooth*. McGraw-Hill Professional. España, 2002.

[5] PalmSource. *Palm OS Developer Suite*. http://www.palmos.com/dev/tools/dev_suite.html.

[6] PalmOne. http://www.palmone.com/.

[7] Stallings W. *Wireless Communications and Networks*. Prentice Hall. 2002.

[8] Tatar D., Roschelle J, Vahey Phil and Penuel W. *Handhelds Go to School: Lessons Learned*. IEEE Computer Magazine. September 2003. pp 30-37.

[9] Viejo Pascuero. "Cuentos Educativos 2002". Chile.

[10] Zambrano J. y Márquez M. *Análisis, Diseño y Desarrollo de un Soporte Didáctico Educativo, Utilizando Dispositivos PDA y el Protocolo de Comunicación Bluetooth*. Publicación interna de la UCV. Mayo 2005.

[11] Zurita G. *Computer Supported Collaborative Learning Activities and Mobile Technology*. PhD Thesis, Pontificia Universidad Católica de Chile, Diciembre del 2003.

Test Rápido de Planificabilidad para R.M. o D.M.

José M. Urriza, Ricardo Cayssials y Javier D. Orozco
Departamento de Ingeniería Eléctrica y Computadoras
Universidad Nacional del Sur / CONICET
8000 Bahía Blanca, Argentina
{jurriza, iecayss, ieorozco} @criba.edu.ar

Abstract

The schedulability test is a necessary analysis that allows designer to guarantee that a real-time system will meet their deadlines during runtime. Since 1973 there have been several methods based on bounds that are necessary conditions as the Liu and Layland bound and the Bini bound in 2001, and exact conditions such as the proposed by Joseph and extended by other authors. However, the exact analysis requires a high computational cost. In this paper, we present an exact schedulability test with a low computational cost that can be implemented online in embedded systems.

Keywords: Schedulability Test, RM, DM

Resumen

El test de planificabilidad es la herramienta necesaria para garantizar que un sistema de tiempo-real pueda cumplir con sus restricciones temporales. Desde 1973, se han desarrollando diversos métodos basados en cotas que garanticen dicha condición necesaria, como la de Liu y Layland, la cota de Bini en el 2001 y análisis exactos, como el desarrollado por Joseph y extendido por otros autores. Pero este análisis exacto conlleva una carga computacional elevada. Este trabajo presenta un test exacto, con una carga de computacional baja que puede ser utilizado en tiempo de corrida en sistemas embebidos.

Palabras Claves: Test planificabilidad, RM, DM

1 INTRODUCCIÓN A LOS SISTEMAS DE TIEMPO REAL

Los sistemas de tiempo real (*STR*) se pueden encontrar en casi todas las ramas de las ingenierías, pero en los últimos años se han extendido a la producción masiva. Los sistemas de control, y comunicaciones son ramas en las cuales más se han desarrollado.

Se puede definir informalmente a un *STR* como aquél que necesita terminar una tarea o trabajo, antes de un determinado tiempo. Estos se caracterizan por poseer entre sus parámetros el tiempo y, conjuntamente con las demás, determinan el resultado o comportamiento del sistema. Formalmente es aceptada la definición de Stankovic[1] que dice que: *en los STR los resultados no sólo deben ser correctos aritmética y lógicamente sino que además, deben producirse antes de un tiempo determinado denominado vencimiento.*

Los *STR* dependiendo del vencimiento de la tarea, se pueden clasificar en tres tipos. El primero no permite que ninguna tarea pierda su vencimiento, por lo cual se los denominan *duros* o *críticos* (*hard*). El segundo permite que se pierda algunos vencimientos, por lo cual se los denominan *blandos* (*soft*). Por último, en la actualidad han aparecido sistemas que permiten sólo una determinada cantidad de pérdidas y a estos se los denominan *firmes* (*firm*), por ejemplo, permiten que al menos 9 de 10 alcancen su vencimiento.

En los *STR duros*, la pérdida de un vencimiento en una tarea puede tener consecuencias adversas. En los sistemas en los cuales la vida humana es participe, por ejemplo la aviónica, se le debe garantizar que todas las tareas terminen antes de su vencimiento mediante un *test de diagramabilidad* o *planificabilidad*. Si el test es exitoso, a estos sistemas se los denomina *diagramables* o *planificables* y se dice que han cumplido con todas sus *constricciones de tiempo*.

En 1973, C. L. Liu y James W. Layland [2] realizan el primer aporte significativo para determinar la diagramabilidad de un *STR-duro*. Este trabajo se ha convertido en un trabajo liminar, el cual ha sido la base para todos los trabajos posteriores. Esto se debe a que formalizan el marco de trabajo necesario para garantizar la diagramabilidad de un conjunto de tareas. En [2], se considera al sistema mono-recurso y multitarea. Se entiende como recurso por ejemplo a un único microprocesador o medio físico de transmisión, el cual sólo puede ser utilizado por una tarea a la vez.

Existen dos formas de planificar las tareas al recurso. La primera forma es predeterminada con anterioridad (estática) y la segunda, que es la más utilizada, se basa en asignar una prioridad a cada tarea.

Una *disciplina de prioridades* es una regla implementada en un algoritmo por el cual, el *diagramador* o *planificador*, determina qué tarea (τ_i) tiene prioridad de ejecución en el recurso. El conjunto de tareas se las consideran periódicas, independientes y apropiables. Una tarea periódica, es aquella que después de un determinado tiempo solicita nuevamente ejecución. La tarea se dice que es independiente cuando no necesita el resultado de la ejecución de alguna otra tarea, para su propia ejecución. Finalmente se dice que una tarea es apropiable cuando el *planificador*, puede suspender su ejecución y desalojarla del recurso en cualquier momento.

Generalmente los parámetros de cada tarea, bajo este marco de trabajo, son: su tiempo de ejecución, que se nota como C_i, su período T_i y su vencimiento D_i. Por lo tanto un conjunto $\mathbf{S}(n)$ de n tareas se encuentra especificado por $\mathbf{S}(n)=\{(C_1, T_1, D_1), (C_2, T_2, D_2),..., (C_n, T_n, D_n)\}$.

Como las tareas son periódicas, el esquema de generación se repite después de un determinado tiempo. Este tiempo se denomina *hiperperíodo* y es el mínimo común múltiplo (*MCM*) de los períodos de las tareas.

En [2] se demostró que el peor esquema de generación, para un mono-recurso, es aquél en el cual todas las tareas solicitan ser ejecutadas en el mismo tiempo y se denomina *instante crítico* o *peor estado de carga*. Se demostró también que, si este estado es planificable, el *STR* es planificable para cualquier otro estado, bajo la disciplina de prioridades utilizada.

El factor de utilización (*FU*) de un conjunto de tareas *duras* $\mathbf{S}(n)$ determina el nivel de utilización del recurso. Este se puede calcular con la siguiente fórmula.

$$FU = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{1}$$

Si el $FU < 1$ se dice que el sistema es *no-saturado*, por lo tanto el recurso tendrá tiempos ociosos en los cuales ninguna tarea requiera ser ejecutada, estos tiempos se denominan *slack* y se podrán utilizar para otros tipos de requerimientos.

Sí $FU = 1$ entonces el sistema se dice *saturado* y no posee tiempos libres.

Sí $FU > 1$ se requiere más tiempo de ejecución que el que se dispone, por lo cual un conjunto de tareas $\mathbf{S}(n)$ con esas características, no es planificable por ninguna *disciplina de prioridades*.

Existen en la actualidad tres *disciplinas de prioridades* básicas para la diagramación de tareas en tiempo real. Estas pueden ser: prioridades fijas, prioridades dinámicas o una combinación de las anteriores. A su vez, las mismas se

diferencian en qué regla implementa el algoritmo diagramador. En prioridades fijas las reglas más utilizadas son Rate Monotonic [2] (*RM*) y Deadline Monotonic [3] (*DM*), en prioridades dinámicas, la más utilizada es Earliest Deadline First [2] (*EDF*) y, por último, una combinación de prioridades fijas y dinámicas es el método Dual Priority [4].

Las condiciones de planificabilidad propuestas en la literatura de tiempo real son complejas y no permiten ser utilizadas en tiempo de corrida debido a la sobrecarga que producen al sistema. Condiciones necesarias han sido propuestas para determinar la planificabilidad de un sistema pero su rigurosidad no cubre muchos sistemas de tiempo real que son planificables. En este trabajo se propone un test de planificabilidad exacto de baja complejidad que puede ser implementado en tiempo de corrida. Se compara la complejidad del test con la de los métodos tradicionales propuestos en la literatura de tiempo real.

El trabajo se organiza de la siguiente manera: en la sección 2 se presenta los trabajos previos. En la sección 3 se presenta el test de planificabilidad que garantiza que el sistema de tiempo real cumplirá con sus restricciones temporales. La sección 4 presenta una heurística para disminuir la complejidad en el test de planificabilidad. En la sección 5 se exponen los resultados experimentales y finalmente en la sección 6 se realizan las conclusiones.

2 TRABAJOS PREVIOS

En [2], Liu presenta una cota que por 13 años fue la única técnica para determinar si un sistema diagramado por *RM* cumplía con sus constricciones de tiempo. Esta cota impone un límite de aproximadamente el 70% (*ln* 2=0,69315) al factor de utilización del sistema cuando el número de tareas tiende a infinito. Esta cota es de simple utilización, dado que si un conjunto de tareas **S**(*n*) con un *FU* es menor a *ln* 2, el sistema es planificable por la disciplina de prioridades *RM*. A continuación se presenta en detalle el cálculo de la cota de Liu:

$$FU \le n.(2^{\frac{1}{n}} - 1) \tag{2}$$

La cota es una condición necesaria pero no suficiente, por lo cual sistemas de tiempo real con un factor de utilización mayor a *ln* 2 pueden ser planificables por *RM*. Por ejemplo: el sistema con periodos 2, 3, 6 y tiempos de ejecución unitarios, es planificable y es *saturado* (*FU*=1).

En 1986, Joseph y Pandya [5] publicaron un trabajo en el cual se definen las condiciones necesarias y suficientes para que un *STR* pueda ser planificable por *RM*. El método propuesto calculaba, partiendo del instante crítico, el peor tiempo de respuesta de cada tarea. Se probó también que no existía una solución analítica de este tipo de problemas y solo era posible calcularlo de manera iterativa. La solución de la ecuación (3) es válida si $R_i^+ = R_i$ y además es $R_i \le D_i$.

$$R_i^+ = C_i + \sum_{j=1}^{i-1} C_j \left\lceil \frac{R_i}{T_j} \right\rceil \tag{3}$$

Diversos trabajos han sido publicados con soluciones similares. En 1987, Lehoczky produce un reporte interno [6], que luego en 1989 es publicado en el *RTSS* de *IEEE* [7]. En 1991 y en 1993, Santos [8, 9] publica dos trabajos de diagramabilidad de *RM* para redes en tiempo-real. También en 1993, Audsley [10] publica un trabajo en el cual, además de dar un test de diagramabilidad similar al de Joseph, somete al *STR* a diversas problemáticas de implementación (eg. jitter, bloqueos, precedencias, etc). Se hace notar que en este trabajo determinan que dependiendo del ordenamiento por prioridades de las tareas (*RM* o *DM*), el test de diagramabilidad desarrollado por Joseph sirve para las dos disciplinas.

En 1982, Leung [3], definió *DM*, pero no se presentó un test de diagramabilidad hasta que fue presentado por Audsley ([11]) en 1991.

En 2001, Bini [12] define una cota hiperbólica que mejora un poco la cota de Liu. Esta se calcula de la siguiente manera:

$$\prod_{i=1}^{n} \left(\frac{C_i}{T_i} + 1 \right) \le 2 \tag{4}$$

Como se puede observar para la cota de Liu [2] y la cota de Bini [12], se necesita calcular el factor de utilización del conjunto de tareas y la productoria respectivamente. Estos cálculos poseen un orden O(*n*). También para la cota de Bini el sistema con periodos 2, 3 y 6 y tiempos de ejecución unitarios no cumple con la cota.

3 TEST DE PLANIFICABILIDAD

En los trabajos previos [5-8, 10, 11], donde se calcula el test exacto, además de garantizar la planificabilidad del *STR*, se obtiene el peor tiempo de respuesta de la tarea para el momento crítico. Para obtenerlo, se necesita que de manera iterativa se busque la solución del sistema. Esta búsqueda convierte a los tests en métodos que poseen una alta carga computacional. Una solución que no busque el peor tiempo de respuesta, pero sí garantice la planificabilidad del sistema tendría una menor carga.

En [13] se presenta un método de *Slack Stealing*, en el cual se determinan en qué tiempos es posible encontrar la solución de las ecuaciones temporales ahí presentadas. Mediante un lema se demuestra que una tarea τ_i, que ha sido retrasada al máximo su ejecución, sólo puede terminar justo antes de que una tarea de prioridad mayor demande ser atendida o antes de que la misma se venza. Por lo tanto se busca el punto donde el Slack es máximo, sin necesidad de estar buscando de manera iterativa cuál es el máximo slack. El conjunto de puntos, donde es posible encontrar la solución se lo denomina $\mathbb{V}_i$. Se presenta a continuación la fórmula del Slack de forma completa y para cuándo es calculado desde $t = 0$.

$$k_i(t^*) = t^* - t - \sum_{j=1}^{i}\left[C_j\left(\left\lceil\frac{t^*}{T_j}\right\rceil - \left\lfloor\frac{t}{T_j}\right\rfloor\right) - c_i(t)\right], \forall t^* \in \mathbb{V}_i$$

$$para \quad t = 0 \qquad (5)$$

$$k_i(t^*) = t^* - \sum_{j=1}^{i} C_j\left\lceil\frac{t^*}{T_j}\right\rceil, \forall t^* \in \mathbb{V}_i$$

Con algunos cambios se puede adaptar este método para realizar solamente un test de planificabilidad.

Los puntos del conjunto $\mathbb{V}_i$ deben satisfacer el lema del trabajo en [13]. Partiendo del peor estado de carga, el intervalo donde se encuentran los puntos de $\mathbb{V}_i$ y donde es posible encontrar un slack ≥ 0 es:

$$[A, D_i] \text{ con } \quad A = \sum_{j=1}^{i} C_j$$

A continuación se presenta el siguiente lema considerando que el sistema **S**(*i-1*) tareas es planificable:

Lema 1:

Sí dentro del intervalo $[A, D_i]$ en los tiempos donde las tareas de mayor prioridad se instancia, o en el vencimiento de la tarea τ_i, el slack para alguno de estos tiempo es ≥ 0, entonces el sistema es planificable por *RM* o *DM* Por el contrario, si todos esos tiempos tienen un *slack* < 0 el sistema es no planificable.

Prueba:

El intervalo de búsqueda está limitado por *A*, y el vencimiento de la tarea τ_i (D_i). Si el *slack* ≥ 0, quiere decir que ha cumplido con todas las constricciones de tiempo del subsistema **S**(*i*). Para el caso contrario, que el sistema no tenga en el intervalo ningún punto con *slack* ≥ 0, el sistema no es planificable, debido a que la ejecución del sistema **S**(*i*) en el intervalo [0, D_i] es sobresaturado. □

El *Lema 1* permite disminuir el intervalo de inspección y consecuentemente reduce la complejidad del cálculo de la planificabilidad.

4 HEURÍSTICA

En [8] se probó que un sistema de tiempo real **S**(*n*) partiendo del peor estado de carga, si posee tiempos ociosos, estos se encontrarán sobre el final del intervalo [0, D_n]. Esto, sumado al *Lema 1* permite construir una heurística de búsqueda de donde es posible que el slack sea mayor o igual a cero. Por lo tanto, la búsqueda se inicia de atrás hacia adelante y sólo es necesario para probar que el sistema es planificable el primer *slack* ≥ 0. La forma en que se realiza es calculando

con la ecuación (5), en primer lugar en el vencimiento de la tarea τ_i y luego en cada invocación de las tareas de mayor prioridad, buscando algún punto con *slack* ≥ 0. A continuación se muestra cómo funciona la heurística comparándola con el método de Joseph en un ejemplo.

Ejemplo:

Sea el siguiente sistema de tiempo real

Tarea τ_i	T_i	C_i	D_i
1	3	1	3
2	4	1	4
3	6	1	6

Tabla 1.

En la figura 1 vemos la ejecución de las tareas en el hiperperíodo.

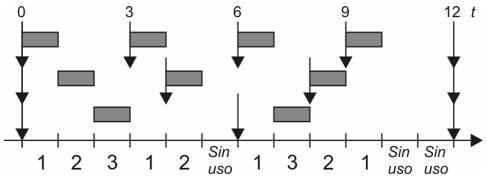


Figura 1.

Como se muestra en la Figura 1, el *STR* es planificable por la disciplina de prioridades *RM*. A continuación se calcula de forma iterativa la planificabilidad por el método de Joseph para dos y tres tareas:

Tarea τ_i	R_i	$R_i^+ = C_i + \sum_{j=1}^{i-1} C_j \left\lceil \frac{R_i}{T_j} \right\rceil$ (3)
2	1	$1+1.\left\lceil \frac{1}{3} \right\rceil = 2$
2	2	$1+1.\left\lceil \frac{2}{3} \right\rceil = 2$
3	2	$1+1.\left\lceil \frac{2}{3} \right\rceil + 1.\left\lceil \frac{2}{4} \right\rceil = 3$
3	3	$1+1.\left\lceil \frac{3}{3} \right\rceil + 1.\left\lceil \frac{3}{4} \right\rceil = 3$

Tabla 2

Ahora se calcula el slack con el valor de *t** en el arribo de las tareas de mayor prioridad o en el vencimiento.

Tarea τ_i	*Intervalo*	*t**	$k_i = t^* - \sum_{j=1}^{i} C_j \left\lceil \frac{t^*}{T_j} \right\rceil$ (5)
2	[2, 4]	4, 3	$4 - \left(1*\left\lceil \frac{4}{3} \right\rceil + 1*\left\lceil \frac{4}{4} \right\rceil \right) = 1 \geq 0$
3	[3, 6]	6, 4, 3	$6 - \left(1*\left\lceil \frac{6}{3} \right\rceil + 1*\left\lceil \frac{6}{4} \right\rceil + 1*\left\lceil \frac{6}{6} \right\rceil \right) = 1 \geq 0$

Tabla 3

Se puede observar que existe una disminución de las iteraciones necesarias para probar que el sistema es planificable. Es esta ventaja que se quiere aprovechar para bajar el costo computacional.

4.1 Complejidad

La complejidad del cálculo, para el caso que el sistema sea no planificable, se puede saber de antemano, dado que es la cardinalidad del conjunto de puntos $\mathbb{V}$.

$$|\mathbb{V}_i| = \sum_{j=1}^{i-1} \left(\left\lceil \frac{D_i}{T_j} \right\rceil - \left\lceil \frac{A}{T_j} \right\rceil \right) + 1$$
$$|\mathbb{V}| = \sum_{z=1}^{i} |\mathbb{V}_z| \qquad (6)$$

5 RESULTADOS EXPERIMENTALES

En esta sección se compara al método desarrollado en [5] con la heurística propuesta. Para esto se generaron tres grupos de 10, 20 y 50 tareas, de manera similar a las generadas en [14]. En los tres grupos los períodos de las tareas se dividieron en tres subgrupos que van de 25 a 100, de 100 a 1000 y de 1000 a 10000 unidades de tiempo.

El primer grupo está compuesto por 4 tareas del primer subgrupo, 3 del segundo y 3 del tercero. El segundo grupo está compuesto por 7 tareas del primer subgrupo, 7 del segundo y 6 del tercero. El último grupo está compuesto por 17 tareas del primer subgrupo, 17 del segundo y 16 del tercero. A su vez los factores de utilización fueron del 70% al 95% en incrementos de 5%. La precisión es de ± 0,5% por factor de utilización.

El período de las tareas fue elegido aleatoriamente con una distribución exponencial. El vencimiento de las tareas se adoptó igual que el período, por ser el peor caso que se puede adoptar. El tiempo de ejecución de cada tarea se eligió aleatoriamente y fue entero.

Se simularon más de 10000 sistemas por factor de utilización y se registró la carga computacional promedio. La forma en que se realizó fue la siguiente: si el subsistema de 2 tareas era planificable se multiplicaba por la cantidad de iteraciones que requirió para comprobarlo y se pasaba al subsistema de 3 tareas y así sucesivamente de manera que se sumó el número de iteraciones requerido por número de tareas, hasta que resultase planificable o no. Luego se obtuvo el promedio del costo de todos los sistemas simulados. La Figura 2 muestra los resultados obtenidos.

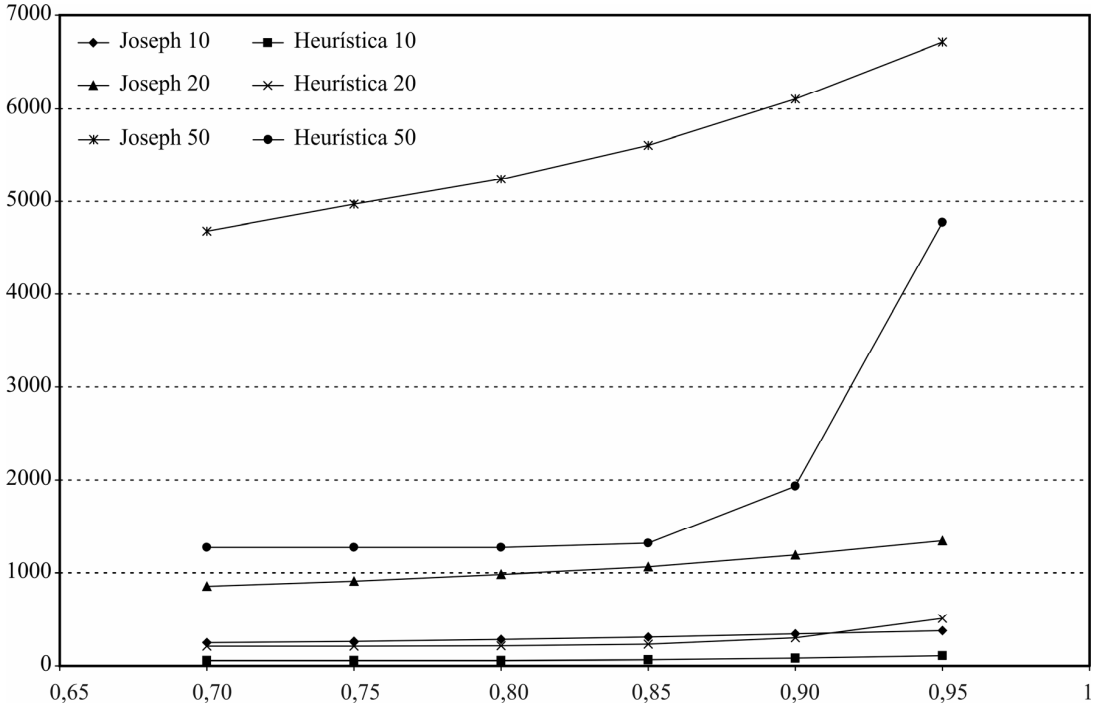


Figura 2. Costo Computacional vs. Factor de Utilización.

Se puede observar que el método aquí presentado obtuvo un menor costo computacional que el método de Joseph para los tres grupos de tareas testeados. Las diferencias son notables para factores de utilización menores al 90% debido

a que la heurística encuentra rápidamente algún punto con *slack* ≥ 0. Para factores de utilización entre el 90% al 100% los dos métodos deben buscar exhaustivamente si los sistemas son planificables, encontrando muchos casos en los cuales no lo son, lo cual incrementa notablemente el costo computacional.

Se resalta el hecho de que para factores de utilización menores al 90% y para los grupos de tareas aquí testeados, en promedio, el valor del costo computacional del cálculo es aproximadamente la suma de la serie:

$$2+3.....+n=\sum_{i=2}^{n} i=\frac{n(n+1)}{2}-1 \tag{7}$$

Esto se debe a que en el cálculo en el vencimiento de la tarea τ_i, el *slack* obtenido es mayor o igual a cero con lo cual, se puede afirmar que el *STR* es planificable y por lo general, se obtiene sólo con una iteración de la ecuación (5).

6 CONCLUSIONES

En este trabajo se presenta un test de planificabilidad de baja complejidad para sistemas de tiempo real. Se comparan los resultados obtenidos con la complejidad de los métodos tradicionales y se comprueba una mejora sustancial mediante del método propuesto.

La complejidad del test permite su utilización en tiempo de corrida en sistemas embebidos para garantizar que las restricciones temporales serán satisfechas cuando el sistema se vea sometido a cambios dinámicos de sus características de tiempo real.

Referencias

[1] J. A. Stankovic, "Misconceptions About Real-Time Computing: A Serius Problem for Next-Generations Systems," *IEEE Computer*, vol. Octubre, pp. 10-19, 1988.

[2] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM*, vol. 20, pp. 46-61, 1973.

[3] J. Y. T. Leung and J. Whitehead, "On the Complexity of Fixed-Priority Scheduling of Periodic, Real Time Tasks.," *Perf. Eval. (Netherlands)*, vol. 2, pp. 237-250, 1982.

[4] R. I. Davis, "Dual Priority Scheduling: A Means of Providing Flexibility in Hard Real-Time Systems," Department of Computer Science, University of York, York, England 1995.

[5] M. Joseph and P. Pandya, "Finding Response Times in Real-Time System," *The Computer Journal (British Computer Society)*, vol. 29, pp. 390-395, 1986.

[6] J. P. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," Department of Statistics, Carnegie-Mellon, Pitsburg, USA 1987.

[7] J. P. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization And Average Case Behavior," presented at IEEE Real-Time Systems Symposium, 1989.

[8] J. Santos, M. L. Gastaminza, J. D. Orozco, D. Picardi, and O. Alimenti, "Priorities and Protocols in Hard Real-Time LANs," *Computer Communications*, vol. 14, pp. 507-514, 1991.

[9] J. Santos and J. D. Orozco, "Rate Monotonic Scheduling in Hard Real-Time Systems," *Information Processing Letters*, vol. 48, pp. 39-45, 1993.

[10] N. C. Audsley, A. Burns, M. F. Richardson, K. Tindell, and A. J. Wellings, "Applying New Scheduling Theory to Static Priority Preemptive Scheduling," *Software Engineering Journal*, vol. 8, pp. 284-292, 1993.

[11] N. C. Audsley, A. Burns, M. F. Richarson, and A. J. Wellings, "Hard Real-Time Scheduling: The Deadline Monotonic Approach," presented at Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software, Atlanta, GA, USA, 1991.

[12] E. Bini and G. Buttazzo, "A Hyperbolic Bound for the Rate Monotonic Algorithm," presented at 13th Euromicro Conference on Real-Time Systems, 2001.

[13] J. M. Urriza and J. D. Orozco, "Métodos Rápidos para el Cálculo del Slack Stealing Exacto y Aproximado para Aplicaciones en Sistemas Embebidos," Dep. de Ing. Eléctrica y Computadoras, Universidad Nacional del Sur, Argentina., Bahía Blanca 2004.

[14] R. I. Davis, K. W. Tindell, and A. Burn, "Scheduling Slack Time in Fixed-Priority Preemptive Systems," *Proceedings of the Real Time System Symposium*, pp. 222-231, 1993.

CRM Shopping Portal for B2C E-commerce based on Relevance Feedback

Giner Alor-Hernández, Pedro Ariza-Acevedo, Lucio Daniel Castelán-Vega, José Oscar Olmedo-Aguirre
Research and Advanced Studies Center of IPN (CINVESTAV),
Electrical Engineering Department. Computing Section,
Mexico City, 07300
{gineralor,pariza, lcastelan}@computacion.cs.cinvestav.mx, *oolmedo@delta.cs.cinvestav.mx*

Abstract

CRM comprises a set of methodologies and practices, usually deployed upon the Internet infrastructure, that help an enterprise to manage customer relationships in an organized manner. It includes all business processes in sales, marketing, and services related to customer needs. A key factor for the CRM success is on providing facilities that assist a customer to define a profile that match her needs with services, products and plans, in order to prevent the customer to remind matters like service details or products previously purchased. Although a number of methods have been proposed for information retrieval where clients provide certain evidence of their needs, oftentimes they have to adjust manually their profiles to match new interests. In this work, we have designed and implemented a CRM shopping portal based on the publish/subscribe pattern that is able to identify those items of interest for a customer by using a group of heuristic rules, generating accordingly adaptive services of the personalized enterprise that the customer requires. The CRM shopping portal provides the means to clients to purchase orders, facilitating the acquisition of products and adapting automatically the heuristic rules used. The main contribution of this work is on considerably improving the recommendations of products and services while maintaining tuned the costumer profile.

Keywords: CRM, e-commerce, information retrieval, relevance feedback.

1. INTRODUCTION

Nowadays, e-commerce has become the most important technological factor to offer services and products from enterprises and institutions to society. E-commerce allows the creation of virtual enterprises that offer their services and products with the aim to attend customer needs. Traditionally, virtual enterprises use products or services catalogs, accessible through Internet. The customers access these catalogs to look for services or products they want to carry out a purchase order. Querying and updating of catalogs is done by the business rules of the enterprise that describe its business policies and activities enactment.

Recently, a tendency has been observed on using ontological structures of concepts that describe the knowledge that has been acquired in an enterprise. To facilitate the information exchange among enterprises, an ambiguity free language that is used to represent ontological knowledge where the terms are chosen by common agreement. In practice, the ontologies have been used to organize the products catalogs in groups, according to the technical features of shared products. Under this approach is possible to formulate heuristic rules on the behavior of the clients' purchase.

In e-commerce, efficient information retrieval about products and services has been a crucial issue in the development of Web-based applications. Different methods have been proposed for information retrieval where the clients provide certain evidence of their interests to outline their needs [1, 2]. Commonly the clients find information about products on Internet by means of search engines such as Google™ and Yahoo™. However, with the use of search engines is difficult to extract specific information due to the vast amount of the returned data, and consequently customers become exhausted, dissatisfied and with wasted of time. Instead of formulating queries and requesting services, it would be more convenient to resort to a notification system that announce to clients when products and services are introduced in the market. In this respect, the request/response pattern from the Client-Server model is inappropriate because the client must formulate the queries explicitly. On the other hand, the search engines are not able to apply

heuristics to filter information because they do not take into account client's profiles. Furthermore in some systems based on the publish/subscribe pattern the input of client's profile is far from been convenient because they are changed manually by filling in forms.

Having this into account, we have designed and implemented a CRM shopping portal based on the publish/subscribe pattern that is able to identify the more relevant aspects of the client's profile by using a group of heuristic rules. By automatically configuring the customer's profile, the virtual enterprise has the means to adapt their services to the customer needs including personalization of Web pages, pre-filing in forms, and shopping behavior history. In this way, the customer gets access to a personalized enterprise that has been remolded by the knowledge that has been acquired from her previous buying behavior. As a result, the CRM shopping portal Web facilitates the acquisition of products or the request of services while maintaining at the same time the customer's profile updated.

The rest of this paper is structured as follows. In the next section we provide the general architecture of the CRM shopping portal. In the following sections we describe the main components and the developed applications of our CRM shopping portal. Next, we describe the functionality of the portal through a case of study. Finally, we review the related work in this area and emphasize the contributions of our work.

2. CRM SHOPPING PORTAL ARCHITECTURE

The CRM shopping portal is based on a layered architecture. Like in other layered architectures, the purpose of each layer is to provide the services required by the upper layers, hiding the details of how the services are implemented. In this sense, each layer has a defined function as explained as follows. Fig. 1 shows the general architecture of the CRM shopping portal.

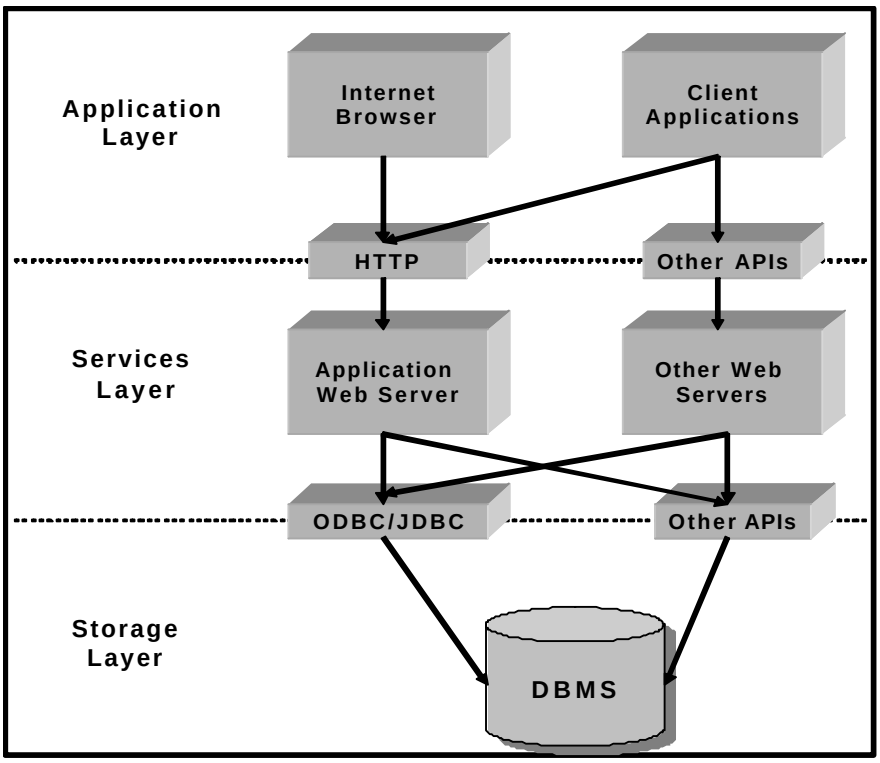


Fig. 1 The general architecture of the CRM shopping portal

Application layer
In this layer, the connections to web pages links and the execution of received functions in JavaScript through a browser is done. Firstly, by using a browser a client formulates HTTP requests to the Web server. Then, the Web server receives the client's requests in order to process them. Next, the Web server sends the response to the client's requests. Finally, the client can visualize this response in a simple and graphical way.

Service Layer or Front end
In this layer, the Web server receives the client's sessions as HTTP requests. The Web server identifies the clients and uses their HTTP requests and presents information managing the required transactions. The services of the CRM shopping portal that interact with the Web server are provided in this layer. All services are represented as hyperlinks within Web pages provided by the CRM shopping portal. As the services are based on the client-server model, only the recommendation service that uses the client's profiles will be explained in the following sections.

Storage layer or Back end
The information coming from the applications in the Web server is stored in this layer. Here, the results of operations insert, delete, modify and update the database are physically performed.

3. CRM PORTAL

In the system, the main components are organized following the Internet client-server architecture:

- The client, identified commonly as the Internet browser, is responsible for interactively choosing the options that the system offers allowing the recovering of prominent information or instantiating any commercial operation.
- The e-commerce server is located inside Web server and is responsible for replying the client's requests.

Fig. 2 represents the functionality of our notification service using the client's profile and the procedures are described among client's requests and Web server's responses. The main page of this service contains a form that inquiries the client to provide her ID and password. The client fills in these fields and sends an HTTP request with the form contents. Our notification system invokes the corresponding application that realizes the verification process in the database building a SQL query. If the query is successful, our notification system now searches for those items that best fit with the purchase profile, in other words, the possible products according the client profile. Here, the notification system considers many factors, using rules and policies to achieve the implementation of heuristics that will retrieve the items mentioned. Information from the database is the basis to conform the client's profile. This information is about previous requests made by the client. Among the policies and rules considered, we can mention:

- Number of times the client buys in the store.
- Quantity of the purchased items by the client.
- Number of times the client buys an item.
- The client's probability in purchasing an item.
- The article's popularity inside an item group.
- The status of an item.
- The article's input date within stock.
- Penultimate and last client purchase date.
- Selection of item groups by their probability.
- Forgetfulness factor.

Once the application has selected the preferred items applying its heuristics, gathers and presents them in the form of catalog and sends them to the browser located in the client side (see Fig.2, step 3).

3.1. Notification process

The client receives the recommended items sent by the server in the form of a catalog where she can select any of them in the desired quantity. If the client wants to make an order, she must send the order contents through HTTP requests to the notification system (step 1 in Fig. 2). Then, the system invokes the corresponding application to carry out the purchase order. The application sends to the client a response containing data for the purchase order reception (step 2 in Fig. 2), in other words, the contact and payment information. Next, the client fills in these fields and sends them with another HTTP request to the Web server; the server receives this parameters and builds a call to the corresponding

application which process the content; the application executes the respective operations such as calculation of the purchase order cost and calculation of delivery time, for mention some, and updates all the database tables related with these processes (steps 3 and 4 in Fig, 2). Finally, a ticket is sent to the client as a successful purchase order (step 5 and 6 in Fig. 2).

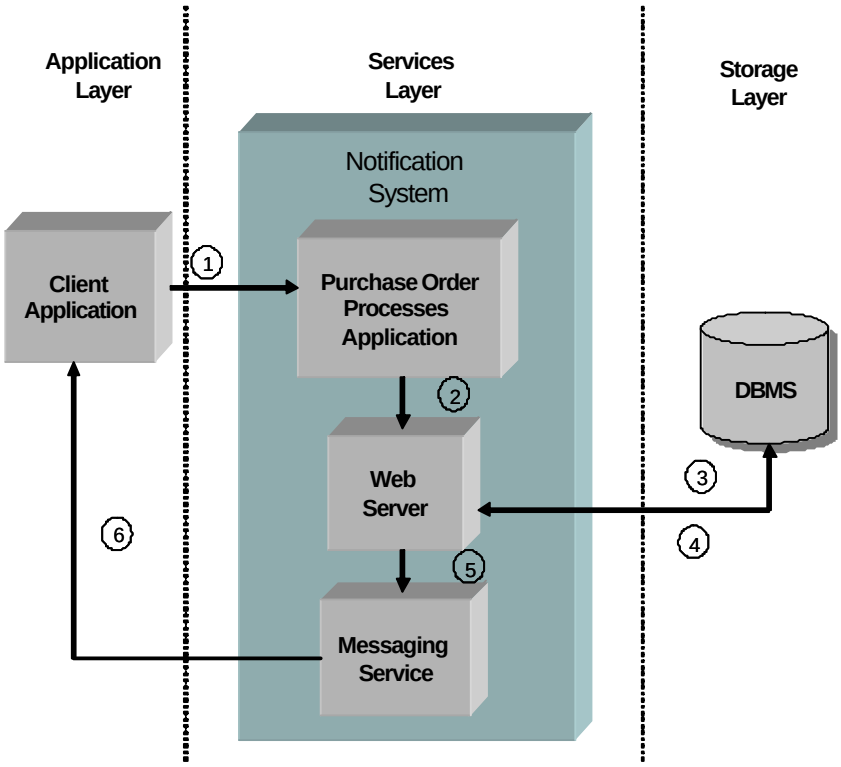


Fig. 2 Notification process by using client's profile

During the purchase order process, some sub processes are generated. These sub processes are described below:

- Calculating the purchase order cost
- Calculating the delivery time
- Generating the invoice
- Building a XML document for the client's credit card agent.
- Generating XML documents for the suppliers if the demanded quantities exceed the stock.
- Continuous interaction with the database for the dynamic updating of the client's profile.
- Generating tickets for the client

Once the ticket is generated, the server sends it to the client and he can visualize it in an Internet browser (Fig. 2, step 5). Fig 3 shows a screenshot of the ticket generated by our notification system.

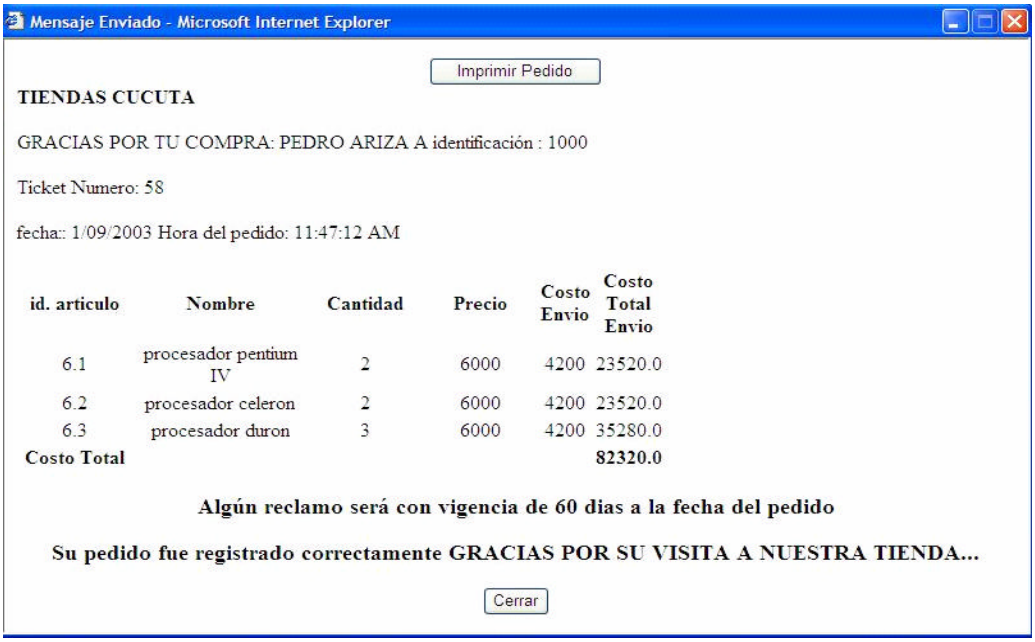


Fig. 3 Screenshot of our notification system

4. MEASURING THE PROFILE QUALITY

The quality of the client's profile is a decisive factor in a CRM system because from its accuracy the system contributes to the client satisfaction through appropriate recommendations. If a vast amount of irrelevant data is returned by the CRM system, the system becomes more problematic than useful. On the contrary, if the system fails in providing to the user enough information, then there is no benefit from its use because the client would continue looking for more information. This issue is called the recall problem. The convenience of the approach adopted to deal with the recall problem will demonstrate the usefulness of the CRM system.

The language used to describe the client's profile is also important. For structured or semi-structured data such as Web pages is more difficult to formulate relational queries due to they involve complicated relations that may lead to return results of unmanageable size. For data based on text, the profiles based on natural language's techniques for information retrieval have demonstrated a reasonable effectiveness to represent the necessities of the users. Nonetheless, clients change very often their preferences as result of the introduction of new products or even due to intense publicity campaigns of previously known products. The inconvenience of the methods that define profiles directly from their clients becomes apparent because they need to enter the new information to maintain consistent their profiles. The approach adopted in this paper solves this problem by inferring the client's preferences from the products they bought in a shopping session. This method requires simply the list of products bought instead of defining a language.

However, even assuming an appropriate representation, the current approaches do not guarantee that profiles provide enough accuracy on recalling. There are two main reasons:

1) The systems generally require that users explicitly specify their profiles, frequently defined as a set of keywords or categories with an associated weight. In such systems is difficult for the users specify in a correct way their information necessities.
2) The most advanced systems of information filtering are generally built assuming that users change their profiles with regularity frequency [3, 4, 5]. If the profile did not maintain at the same time the information necessities of the user, the accuracy and recall become great issues. However, the solution provided in this paper, in spite of their simplicity, does not solve the problem of using a unique vector. In other approaches, the system represents the interests of the users in terms of unique vector (or independent vectors) [3, 5]. The unique vectors are insufficient to model the interests appropriately producing values with low effectiveness.

The use of multiple vectors provides more effectiveness, although in the current systems, the multiple vectors are managed independently producing redundant storage, duplicated processing and the necessity of handling more information on the specifications of necessities.

4.1. Maintenance of the Client's Profiles

Some publish/subscribe systems require that clients manually specify in their profile any change in their interests. This approach delegates to the client the responsibility of identifying and making changes in the profile. In comparison, other approaches use autonomous mechanisms based on a technique called relevance feedback [3, 6] where the clients provide only certain evidence of their interests to adjust their profiles.

One of the most accepted representations for client's profiles is the vectorial space model. In the vectorial space model [7], documents with textual content are represented as vectors in an N-dimensional space where N is the number of words or different terms that appear in the document. The documents that describe similar topics are those that appear closer in this space because it is very likely that they include common terms. Using this model, a profile can be represented as a vector or group of vectors that indicate the client's preferences. In general, a profile vector is near to other when it contains common profiles. The effectiveness of this notion of distance is measured by its accuracy. The accuracy is the percentage of documents obtained that are relevant. The recalling effectiveness is the percentage of produced relevant documents among all the retrieved documents. Both metrics are inversely proportional because the ideal effectiveness is obtained when all the documents are retrieved (including those that are not relevant), in this case the accuracy would be minimum. In the vectorial space model, each document is represented by a vector of pairs of terms and corresponding weights. If there are N different terms in a document D, then D is represented by a vector V:

$$V = [(T_1, W_1), (T_2, W_2), \ldots, (T_n, W_n)] \quad \textbf{(1)}$$

In general, a term T_i is a word occurring in the document and its weight W_i is a measure of the relative importance of the term in the document. The vectorial representation of a document requires the elimination of stop-list [8], i.e. the list of non-discriminating words. The weight of a term is calculated by the term frequency-inverse document frequency (TF-IDF):

$$W_{t,d} = TF_{t,d} \, Log_2 (N/IDF_t) \quad \textbf{(2)}$$

Where:

$W_{t,d}$	is the weight of the term T within document D
$TF_{t,d}$	is the frequency of the term T within document D
IDF_t	is the number of documents that contains term T
N	is the total number of documents in the collection

Length normalization is used to manage documents with different lengths and consists on dividing the terms weight by vector length. The angle between two vectors gives an effective measure of the similarity of the documents as profiles:

$$Cos (V_1, V_2) = V_1.V_2 / |V_1| \, |V_2|$$

with

$$V_1.V_2 = \Sigma_t \, W_{t,d1} \, W_{t,d2}$$

$$|V| = \sqrt{(V \cdot V)} = \sqrt{(\Sigma_t \, W_{t,d1} \, W_{t,d2})}$$

where $V_1.V_2$ is the inner product from $V_1.V_2$ and $|V|$ is the vector length.

In this model, the similarity is defined as the inner product of vectors. For example, when two documents have different content, there will be terms that are included in one document (having non null weight) but not in the other (having null weight) so that $\Sigma_t \, W_{t,v1} \, W_{t,v2}$ is nearly zero.

In our CRM model, profiles are represented as vectors in an *N*-dimensional space where *N* is the number of different products. The profiles that describe similar interests are close in this space because it is likely that share common preferences. Under this interpretation, the popularity IDF_t / N of a product *T* in the collection of *N* purchase invoices is the number IDF_t of times a product occurs in those invoices.

The frequency $TF_{t,d}$ of a product *T* in an invoice *D* is the proportion of the number of times that the product *T* has been bought among the total number of products bought in the invoice *D*. The frequency of the item in the profile can be compared to the proportion of the number of times that the client has visited a store to buy the product with the total number of times that he has visited the store to buy at least an item. Under this interpretation, a high frequency indicates that the client maintains his interest in that product.

However the TF-IDF model [8] is not complete adequate because the time factor is not considered in the product frequency in a profile. For example, the same frequency can be obtained from a client who has bought a product with regularity than form another that bought the product at first but that does not buy it anymore. Therefore, a CRM system should only recommend the product to the first client. Our CRM system considers the time factor based on the last date of client's purchase. In dynamic environments, a CRM system should revise the product's relevance in the clients' profiles formulating better recommendations. On the other hand, to simplify the weight's evaluation from the items with the purpose of reducing the time implied in the involved transactions, in this paper we have defined the weight factor as purchase probability. Where the purchase probability is defined by P = A/G, where:

A is defined as the number of times the client has bought that item, and
G is defined as the total number of times the client has bought in the store (i.e. there is an invoice for that purchase).

In this respect, we describe our approach based on relevance feedback as explained in as follows.

4.2. Our approach based on Relevance Feedback

Relevance feedback is an effective technique for retrieving information based on the documents content [7]. The main idea uses the document that has been evaluated already by the user, considering the terms that appear within the document such as relevant while those considered in future queries of the same query are not relevant.

$$Q_{i+1} = \alpha Q_i + \beta \sum_{d \in R} V_d - \gamma \sum_{d \in NR} V_d$$

Where:

Q_i	is the initial vector from a query,
Q_{i+1}	is the modified vector,
V_d	is the vector representing the document *D*,
R	is the relevant document collection,
NR	is the non-relevant document collection, and
α, β and γ	are fixed parameters of the used feedback method.

The method defined in [3] is a well-known effective schema that instantiate these parameters to $\alpha = 1$, $\beta = 2$ and $\gamma = 0.5$. In our approach, instead of using an exponential function to incorporate a forgetfulness factor (times the client has not bought a product), we use the last purchase date as control parameter, being the main objective to identify and to select the items of a client's profile using the recommendation service. The client's profile model given in (1) is used in our CRM system to demonstrate the accuracy in the recommendations. Furthermore, our CRM system takes into account the changes in the client's profile. As explained next, some heuristics are used in our CRM system to keep the item's profile every time that a purchase is done, filtering information and avoiding duplicity. In Fig 4, this process is shown.

Each heuristic uses a vector as specified in (1) where T_n is the selected item and W_n is the given weight for that selection. Each weight is calculated depending on the logical result obtained by a combination of conditions in each heuristic. The weight is calculated depending on the Boolean operation applied to each weight W_n. If the operation result becomes 1 (Boolean value), the product belongs to the profile set of selected items by the client, whereas the result becomes 0 (Boolean value), the item is rejected and not included.

In Fig 4, our notification system applies the following four heuristics:

1) The preferred items according to its preference, i.e. those items that the system believes the client would buy according to his previous purchases if it exceeds a predetermined threshold;
2) The items considered as similar to those chosen by heuristic 1;
3) The items belonging to a group of those chosen by heuristic 1;
4) The most popular items in that moment within the whole store and according to those chosen by heuristics 1, 2 and 3.

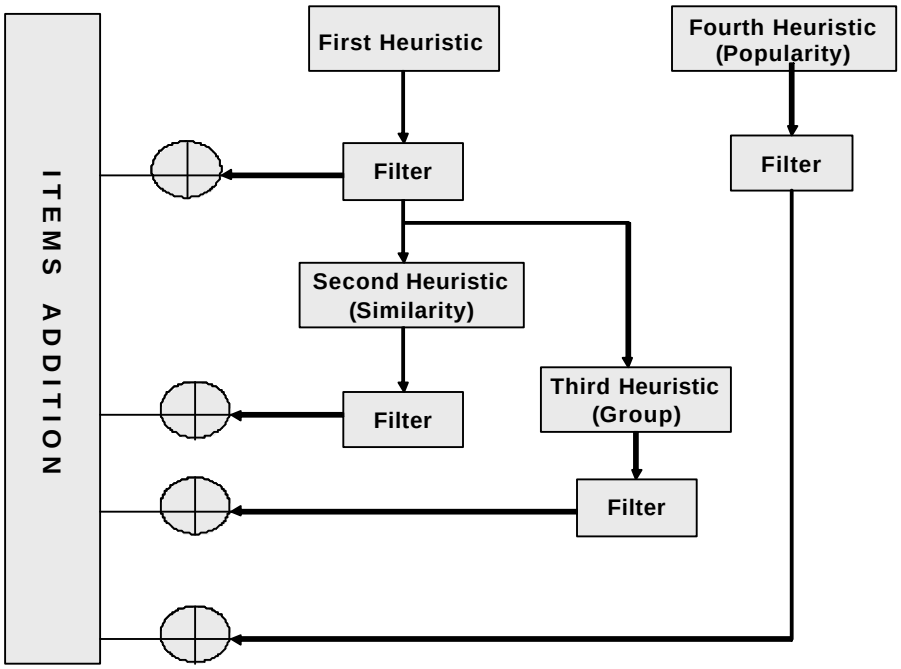


Fig. 4 A general functionality from four heuristic rules used in the CRM shopping portal

These heuristics are applied every time the client makes a purchase. Then, the notification system adjusts the clients profile according to the selected products and generates a list of preferred items from the client's requests. The heuristics are used with prominent information from the last purchases, the date in that the product entered into the stock, an estimated forgetfulness factor and threshold range. Although, each item may be obtained by applying a heuristic some filters are applied to avoid duplicity among items. However, none of these criteria are used if the items are not in stock.

With the following example, we show the operations used to calculate the weight W_n of each product.

Suppose that $W_n = (A * B) + C$, where the operation + represents a disjunction OR and the operation * represents a conjunction AND. Then, substituting the values $A = 1$, $B = 0$ and $C = 1$ in the expression $(A * B) + C$, the evaluation gives W_n as a result = 1, indicating that the evaluated item should be included in the items lists of the client's profile. The active and inactive states are used for the selection of the item's profile that is considered as an excellent evaluation approach by obtaining these items. For the second, third and fourth heuristic a small threshold of 0.3 was evaluated regarding the purchase frequency. The heuristics used by our CRM system are:

1. Previous purchases
2. Similar products
3. Products group
4. Popularity-based products

Next, we described each one:

1. Previous purchases. Based on previous purchases, our CRM system recommends certain customized products using statistical data.

$$V_1 = [(T_1,W_1), (T_2,W_2), ..., (T_n,W_n)]$$

Where:

V_1 is the vector that stores items obtained from previous purchases.
T_i is the selected item, given $1 >= i <= n$.
W_i is the weight given to the item, obtained by using the following formula:

$$W_i = A * B$$

where:
A is related with the purchase probability from an item. The purchase probability from an item is defined as the quotient of the number of times that the item has been bought by the number of times that he has bought in the store. The product must be in the list if it is over the threshold according to its probability, if (probability>=0.5), then A= 1, otherwise A= 0.
B is the relationship with the previous purchase done by the client (a limit is managed, but the CRM system leaves this option open). When the purchases were made before than 90 days, then $B = 1$, otherwise $B = 0$.

2. Similar products. With customized products, our CRM system adds other available products that are similar to them. The similarity between two products is defined by the membership to a common category according to products ontology. For example, suppose that products A and B belong to category C. If the client has bought product A, our CRM system will also recommend product B. In other words, these products are selected because they belong to the same category but they have a recent impact to the system's stock. The advantage of this rule consists on helping the client to identify products that are unknown for her. The selected items recommended from the first heuristic are taken as input for this heuristic.

If T is a generated item by using the first heuristic, then:

$$V_2 = [(T_1, W_1), (T_2, W_2), ..., (T_n, W_n)]$$

Where:

T_i is the selected item with $1 <= i <= n$, where $T_1 ... T_n$ is in the same category or family from T. If $T \in F$ where F is a family, then $T_1 ... T_n \in F$.
V_2 is the vector that stores the products by similarity.
W_i is the given weight to the product and is obtained by using the following formula:

$$W_i = C$$

Where:
C is related with the date the item T_i entered into the stock. A range is considered in comparison with the current date (our CRM system leaves this range open). If the date is not before 60 days, then, $C = 1$, otherwise $C = 0$.

3. Products group. If the client selects an item that belongs to a group of products that are generally sold together, our CRM system recommends the other products in the group. The group is defined by statistical evidence that shows a correlation in sells through historical data. For example, assuming that there is statistical evidence that products A, B and C have been generally sold together, if the client selects product A, our CRM system will recommend B and C products. Furthermore, the system takes care of avoiding redundant recommendations. The CRM system prevents a redundant recommendation of product C, when the client selects products A and B. The advantage of incorporating this rule consists on reducing the number of selections that the client makes.

If T is a generated item by using the first heuristic, then:

$$V_3 = [(T_1, W_1), (T_2, W_2), ..., (T_n, W_n)]$$

Where:

T_i is the selected item with $1 >= i <= n$, where $T_1 ... T_n$ belong to T. If $T \in G$ where G is a group, then $T_1 ... T_n \in G$.
V_3 is the vector that stores the items by groups.

4. Based-popularity products. Our CRM system recommends products according to their popularity. Any product in stock has an indicator of its popularity that is relative to the category the product belongs and in comparison to all kind of products. The popularity allows to the CRM system to be able to select products that have great acceptance in that moment.

$$V_4 = [(T_1, W_1), (T2, W_2), ..., (T_N, W_N)]$$

Where:

V_4 is the vector that stores the items by their popularity
T_i is the selected item
W_i is the weight given to the product by using the following formula:

$$W_i = D + E$$

Where:

D is related with the popularity of the item T_i corresponding to its category or family. The popularity of an item is defined as the proportion of the number of times the client has bought that item with the number of times number that he has bought products belonging to the same family. If the popularity >=0.3, then $D = 1$, otherwise $D = 0$.

E is related with the popularity of the item W_i corresponding to whole products. E is defined as the proportion of the number of times the client has bought that item, with the number of times that he has bought products in a general way. If popularity >=0.3, then $E = 1$, otherwise $E = 0$.

The generation of the initial data for the client's profile is a well-known problem on information retrieval because there is no evidence of preference about any product. In our CRM system, the initial values of the profile are taken directly from the first purchase. In this way, the management of the profile is hidden to the client, making unnecessary the amount of information and effort involved in the use of traditional techniques.

5. RELATED WORK

The research community of information retrieval has dedicated considerable efforts to propose new user's profiles with textual representation, especially in routing and TREC filtrate [9]. Under the routing approach, a set of documents whose content address a topic is given to a CRM system to carry out sorting, according to evidence based on the built profile. Under the filtering approach, a CRM system is requested to make a decision whether the document is relevant or not. The main emphasis of the TREC tasks is its acceptable performance.

Publish/subscribe protocols have widely been used in the database and information processing communities. Recent projects as Stanford's C3 [10], OGI's continuous queries [11] and Grand Central of IBM Almaden [12] provide a management component of the user's profile. However, those projects have not emphasized the learning based on the profiles acquisition and maintenance. The automated learning community has shown great interest on the different profiles generation aspects, especially in customized information filtering [13]. In [14], the relevance feedback in environments for information filtering is studied. They demonstrate effectiveness on the obtained results when some documents are analyzed each time. In [15], the effectiveness of the incremental relevance feedback is evaluated, but from a point of view of information retrieval environments. In [16], an approach based on descent gradient for CRM systems with textual information is proposed. They used categorized profiles to represent user's profile and a comparison against the Rocchio and Balabanovic classic methods was done obtaining better results. In [17], the Latent Semantic Indexation (LSI) is proposed to derive a reduced vectorial space and built profiles vectors for each analyzed document by the user. The relevance of a document with respect to its profile is derived based on the similarity (cosine) to the nearest profile vector.

In [3], an approach is proposed using publish/subscription schemas for the information dissemination in wide areas. This approach requires that users upload and update their profiles using relevance feedback.

6. CONCLUSIONS

B2C ordinarily refers to on-line trading and auctions, for example, on-line stock trading markets, on-line auction for computers and other goods. B2C e-commerce refers to the emerging commerce model where businesses /companies and clients interact electronically or digitally in some way. In this paper, we have proposed a mechanism based on four heuristics that improve the relationships among clients implementing business rules based on data feedback by the client's previous purchases. Our CRM system for B2C e-commerce is based on the client's profiles acquired during their previous purchases; therefore it allows improving future recommendations of products and services.

References

[1] Balabanovic, M., An Adaptive Web Page Recommendation Service. In Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, CA, February 1997.

[2] Rocchio, J.J., Relevance feedback in information retrieval. In The Smart System – Experiments in Automatic Document Processing, pp. 313-323, Prentice-Hall 1971.

[3] Yan, T.W., García-Molina, H., SIFT - a tool for wide-area information dissemination. In Proceedings. of the 1995 USENIX Technical Conference, pp. 177-186, 1995.

[4] PointCast, http://www.pointcast.com/. 1989

[5] MyExcite Channel. http://my.excite.com/. 1999.

[6] Salton, G., Buckley, C., Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science, 41(4):288-297, 1990.

[7] Salton, G. Automatic Text Processing, Addison-Wesley, 1989

[8] Frakes, W.B., Baeza-Yates, R., Information Retrieval: Data Structures and Algorithms. Prentice-Hall, 1992.

[9] Voorhees, E.M., Harman, D., Overview of the fifth Text Retrieval Conference, (TREC-4). In The Fifth Text Retrieval Conference (TREC-95), NIST, Gaithersburg, 1996.

[10] Chawathe, S.S., Abiteboul, S., and Widom. J., Representing and querying changes in semi-structured data. In the International Conference on Data Engineering. Orlando, 1998.

[11] Liu, L., Pu, C., CQ: A personalized update monitoring toolkit. In the ACM SIGMOD Conference. Seattle, 1998.

[12] IBM, Information on the fast track. IBM Research Magazine, 35, 1997.

[13] Sheth, B., Maes, P., Evolving agents for personalized information filtering. In Proceedings of the Ninth Conference on AI for Applications, pp.345-352. IEEE Computer Society, 1993.

[14] Allan, J. Incremental relevance feedback for information filtering. In the Proceedings of the ACM SIGIR Conference, Zurich, Switzerland, August 1996.

[15] Aalsbersberg, I.J., Incremental relevance feedback. In Proceedings of the ACM SIGIR Conference, pp. 11-22, 1992.

[16] Balabanovic, M., Exploring versus exploiting when learning user models for text recommendation. User Modeling and User-Adapted Interaction, 8(1), 1998.

[17]Foltz, P.W., Dumais, S.T., Personalized information delivery: An analysis of information filtering methods. Communications of the ACM, 35:51-60, 1992.

An Evaluation Mechanism for Procedural SQLf

Marlene Goncalves
Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
mgoncalves@ldc.usb.ve

and

Leonid Tineo
Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
leonid@usb.ve

Abstract

Traditional Database Systems are based on Classic Logic and Set Theory, limiting the expression of user preferences. Fuzzy Logic has been proposed for information representation and manipulation in databases, being the current tendency to study mechanisms of Flexible Querying for Traditional Databases. SQLf has been one of the most remarkable efforts in this tendency. Several evaluation mechanisms have been proposed for SQLf in order to obtain reasonable processing time of fuzzy queries. We have extended SQLf with procedural sentences including fuzzy logic features: fuzzy control structures, fuzzy anonymous blocks, fuzzy stored procedures, fuzzy functions and fuzzy triggers. For evaluation such sentences we propose here the application of the Derivation Principle based mechanism. This mechanism consists in the distribution of the λ-cut over conditions involved in fuzzy queries in order to keep low the number of row access in fuzzy sentence evaluation. This principle has been used before for other kind of sentences in SQLf, shown the best performance between known evaluation mechanisms for fuzzy queries.
Keywords: Database query processing, Query Languages, SQLf, Procedural Language.

1. INTRODUCTION

Boolean logic restricts the treatment of imprecision or uncertainty in database systems. A solution for this problem is using fuzzy sets in databases systems in order to express non-crisp data and gradual (or flexible) requirements. There are some different proposals of flexible querying languages based on fuzzy sets; SQLf [4] is one of them. This language has been proposed by IRISA-ENSSAT research team, whose leader is Patrick Bosc.

In previous works, we have studied the new features of SQL2 [1] and SQL3 [12] determining which of them are susceptible of a fuzzy treatment. SQL2 norm incorporates relational algebra operations, mechanisms for constraints specifications and subqueries in the FROM clause. SQL3 norm incorporates some features of deductive databases, active databases and object oriented databases. Previous works leaded us to the definition of two extensions of SQLf, named: SQLf2 [8] and SQLf3 [9], corresponding to the definition of fuzzy querying components based on SQL2 and SQL3 norms, respectively. Due to SQL3 includes a procedural language, our SQLf3 also includes a fuzzy procedural language, that is, a language that contains procedural sentences with fuzzy extensions.

The use of flexible querying languages supposes a great amount of computation due to the satisfaction degree calculation and the fact that the fuzzy answer involves more rows than result of similar crisp queries. On the other hand, usual access path that ensures good answer times in regular queries is not obvious for fuzzy querying systems. Nevertheless, Bosc et al. have proposed an evaluation mechanism that attempts to keep low the number of row access in fuzzy query processing by means of relationship between fuzzy sets and crisp ones. Such relation is established by the fuzzy set operator λ-cut. This operator defines the crisp set containing all elements of the fuzzy set with membership degree greater or equal to λ. As SQLf allows user to specify a minimum satisfaction degree of the desired rows, λ-cut concept may be used in order to process the fuzzy query by means of the result of a regular one computing λ-cut of the answer [3]. This evaluation mechanism is called Derivation Principle Strategy.

In this paper, we propose the application of the Derivation Principle to fuzzy procedural sentences of SQLf3 in order to define a low cost evaluation mechanism for this kind of sentence.

2. LANGUAGE DEFINITION

SQLf is the most complete fuzzy extension to SQL [13] due to the diversity of fuzzy queries that allows the extension of all SQL constructions based on fuzzy sets use. Fuzzy queries involve fuzzy terms (atomic predicates, modifiers, connectors, comparators and quantifiers) whose meaning depends of the user and the context. The result of a query in SQLf is a fuzzy set. In SQLf, the querying basic block is:

SELECT <attributes> FROM <relations> WHERE <fuzzy condition> WITH CALIBRATION [n|λ|n,λ]

Semantic of this construction is Cartesian product of the relations involved in the FROM clause, selecting those rows that satisfied the fuzzy condition and taking the fuzzy projection of those attributes indicated in the SELECT clause. In the WHERE clause, some logical expressions can be used with user-defined terms and predefined operators of fuzzy logic. In the WITH CALIBRATION clause, a tolerance to select rows is specified. This process is called calibration. In the first SQLf definition due to Bosc and Pivert [4], the calibration was specified in the SELECT clause, in later works [8][9], we have extended the definition with this clause in pro of language orthogonallity. Calibration may express a maximum number *n* of best elements that must be returned in the answers. It is called a quantitative calibration. There is also a qualitative calibration. This later consists in specifying a satisfaction degree λ; rows in the result must have satisfaction degree greater or equal to λ.

Inspired in procedural sentences, we have defined SQLf3 that includes procedural language features with fuzzy logic treatment [9]. One or more blocks compose each procedural unit in SQLf3. A block can be nested in another block. Program types used in SQLf3 are:

Anonymous Block: A program block may be defined and run into a SQLf shell. This kind of programs is often called Anonymous Block. The blocks contains three sections:

— The declaration section, is the section that includes variables, cursors and user-defined exceptions. This section is optional. It is identified by DECLARE word.

— The actions' section is formed by a sequence of fuzzy procedural statements. This section is mandatory. It is identified by BEGIN word.

— The exception handling section includes actions that must be performed when an exception is raised. This section is optional. It is identified by EXCEPTION keyword.

The syntax of an anonymous block is:

[DECLARE {<variable definition>|<cursor definition>|<exception definition>}]
BEGIN <SQLf>|<fuzzy_procedural_sentences> [EXCEPTION <actions>] END

Stored Procedures: The stored procedures contain SQLf and fuzzy procedural statements. A procedure can ever be executed repeatedly and it accepts parameters. The syntax of a stored procedure is

CREATE OR REPLACE PROCEDURE <procedure_name>([<parameter_name>
IN|OUT <datatype> [,<parameter_name> IN|OUT <datatype> ...])
IS <variables>|<cursors>|<user-defined-exceptions>
BEGIN <SQLf>|<fuzzy_procedural_sentences> EXCEPTION <actions> END

Fuzzy Functions: The fuzzy functions defined by the user contain SQLf and fuzzy procedural statements. A fuzzy function can be executed repeatedly and it accepts parameters. The syntax of a fuzzy function is:

CREATE OR REPLACE FUNCTION <function_name>([<parameter_name>
IN|OUT <datatype> [,<parameter_name> IN|OUT <datatype> ...])
RETURN <datatype> IS <variables>|<cursors>|<user-defined-exceptions>
BEGIN <SQLf>|<fuzzy_procedural_sentences> RETURN <value>;
EXCEPTION <actions> END

Triggers: A fuzzy trigger is associated with a table and it is executed automatically when updating sentences are executed. Its components are:

— The fuzzy action can be executed before, after or instead of triggering event.
— The fuzzy action can refer to either new or old values of rows that were inserted, eliminated or updated.
— Update events may specify a particular or set of columns.
— A fuzzy condition can be specified by a WHEN clause, and the fuzzy action is only executed if the rule is triggered and the fuzzy condition is true when the triggering event occurs. The fuzzy condition has associated a calibration degree.
— The programmer has the option of specifying the fuzzy action is performed either once for each modified row or once for all the rows that are changed in an operation.

The syntax for the creation of a trigger is:

CREATE TRIGGER <name> ... WHEN (<fuzzy_condition>)
[WITH CALIBRATION λ] {<SQLf>}

Procedural Statements: Extension of the language procedural with new control instructions (case, if, loop, for), including conditions and fuzzy instructions. SQL commands that are used with the control instructions will be able to include fuzzy conditions and instructions. The syntax for each one of the control instructions are:

— If Statement:

IF <fuzzy condition> THEN {<SQLf>} [{ELSEIF <fuzzy condition>
THEN {<SQLf>}}] [ELSE {<SQLf>}] [WITH CALIBRATION λ]END IF

— Case Statement:

SELECT CASE(<attribute>) [WITH CALIBRATION λ]
{WHEN <fuzzy predicate> THEN <action>} END
FROM <tables> WHERE <fuzzy_condition>

— Loop Statement:

LOOP {<SQLf>} END LOOP WHILE <fuzzy condition> DO {<SQLf>}
[WITH CALIBRATION λ] END WHILE

— Repeat Statement:

REPEAT {<SQLf>} UNTIL <fuzzy condition>
[WITH CALIBRATION λ] END REPEAT

— For Statement:

FOR result AS <SQLf> DO {<SQLf>} END FOR

3. Derivation Schemes

The Derivation Principle Strategy is a fuzzy query evaluation mechanism that attempts to keep low the number of row access in fuzzy query processing by means of the distribution of the λ-cut over conditions involved in the fuzzy query. It derives a regular SQL query whose result contains the rows satisfying the fuzzy query with a degree at least .λ. This principle may be used for the evaluation of fuzzy queries avoiding the scanning of the whole database. In some cases the

derived query produces a superset of the λ-cut. The derivation is said to be strong when it produces just the λ-cut, if there are extra rows, the derivation is weak. The application of this principle to SQLf querying structures has been object of previous works [2][3][4][5][8][9][10][11][14]. We restrict our work to fuzzy sentences that allows strong derivation.

We denote the regular sentence obtained from the given fuzzy sentence with qualitative calibration, applying the derivation principle as: DS(<fuzzy sentence>). Either for subqueries or nested sentences, the calibration is inherited from the outer level sentence. In this case we use the notation: DS(<subsentence>, ≥, λ).

A basic concept in derivation principle application is the called "derived necessary condition". It is a regular condition that expresses the λ-cut of a fuzzy condition. We denote it by: DNC(<fuzzy condition>, ≥, λ).

In order to illustrate this principle, let's show it with a basic block query. Let *location* be a relation with *priority* attribute, and *BestZone* a fuzzy predicate defined on the priority attribute domain by the membership function in Fig. 1.

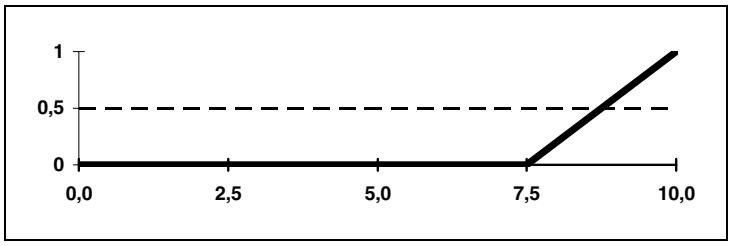


Fig. 1. BetterZone predicate definition by means of their membership functions. We remark the satisfaction 0.5 that is used as a threshold in the example

The user requirement: *Find those locations in best areas with a tolerance level of 0.5* may be expressed as:

FQ: *SELECT * FROM location WHERE priority=BestZone WITH CALIBRATION 0.5*.

Applying the derivation principle, we obtain:

$$DNC(priority = BestZone, \geq, 0.5) = Priority \geq 8.75.$$

Then:

$$DS(FQ) = SELECT * FROM\ location\ WHERE\ priority \geq 8.75.$$

This query is the same that

$$DS(SELECT * FROM\ location\ WHERE\ priority = BestZone, \geq, 0.5).$$

Using the derivation principle to SQLf procedural sentences, we may also obtain regular sentences that would be the base for the evaluation of the fuzzy construction. In the following, we will show the derivation schemes (transformations) for considered fuzzy procedural sentences. In [7], we have proved presented transformations by logic derivation using the definition of the querying operators. We do not present here proofs due to space restrictions. We must remark that we have focused our work to queries involving fuzzy sentences allowing strong derivation (that is, we may derive a regular query expressing just the λ-cut of the fuzzy query).

We will illustrate these statement transformations with several examples. All of them are based on *flights* table with *origin*, *destination*, *airline*, *number*, *departure_time*, *arrival_time* and *price* attributes, and fuzzy predicates: *low*, *medium*, *high* and *good* defined by Fig. 2 and Fig. 3.

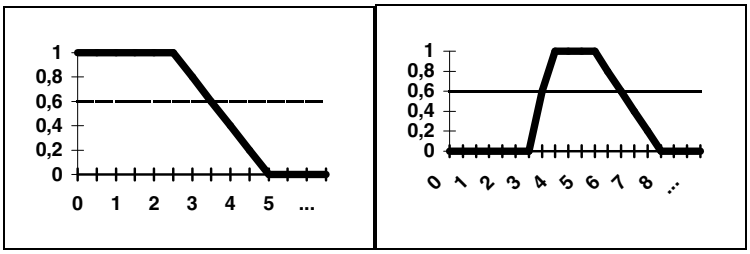


Fig. 2 Fuzzy predicates definition by means of their membership functions. Left: *low* predicate on time differences. Right: *medium* predicate on time differences. We remark the satisfaction 0.6 that is used as a threshold in the example.

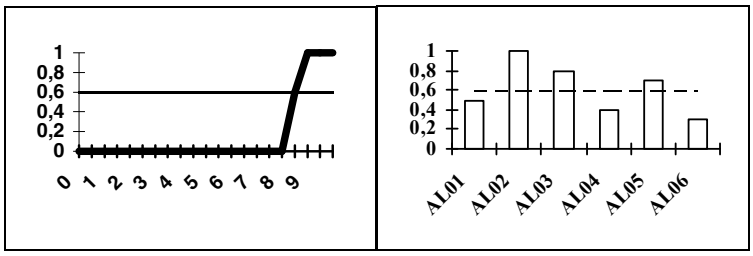


Fig. 3 Fuzzy predicates definition by means of their membership functions. Left: *high* predicate on time differences. Right: good predicate on Airlines. We remark the satisfaction 0.6 that is used as a threshold in the example.

The derived sentences for the statements considered in this work are presented in the following subsections.

3.1 Anonymous Block

DS(
DECLARE <variables>|<cursors>|<user-defined-exceptions>
BEGIN
<SQLf>|<fuzzy_procedural_sentences>
EXCEPTION <actions>
END
)=
DECLARE <variables>|DS(<cursors>)| DS(<user-defined-exceptions>)
BEGIN
DS(<SQLf>)| DS(<fuzzy_procedural_sentences>)
EXCEPTION DS(<actions>)
END

For example, the user requirement Increase prices in 10% of those flights whose airlines are good, with a threshold of 0.6, may be expressed as:

FQ =
BEGIN
*UPDATE flights SET price = price*1.1*
WHERE airline = good
WITH CALIBRATION 0.6;
END;

The corresponding derivation is:

DQ(FQ) =
BEGIN
*UPDATE flights SET price = price*1.1*
WHERE airline IN {'AL02', 'AL03', 'AL05'}
END;

3.2 Stored Procedures

DS(
CREATE OR REPLACE PROCEDURE <procedure_name>
([<parameter_name> IN|OUT <datatype>
[,<parameter_name> IN|OUT <datatype> ...]]) IS
<variables>|<cursors>|<user-defined-exceptions>
BEGIN
<SQLf>|<fuzzy_procedural_sentences>
EXCEPTION <actions>
END
) =

```
CREATE OR REPLACE PROCEDURE <procedure_name>
([<parameter_name> IN|OUT <datatype>
[,<parameter_name> IN|OUT <datatype> ... ]]) IS
    <variables>|DS(<cursors>)|DS(<user-defined-exceptions>)
BEGIN
    DS(<SQLf>)|DS(<fuzzy_procedural_sentences>)
    EXCEPTION DS(<actions>)
END
```

The user requirement Increase prices in 10% of those flights whose airlines are good and nPrice is a parameter for price, with a threshold of 0.6, may be expressed as:

```
FQ =
    CREATE PROCEDURE increase_price (nPrice NUMBER)
    BEGIN
        UPDATE flights SET price = price*1.1
        WHERE airline = good and price = &nPrice
        WITH CALIBRATION 0.6;
    END;
```

The corresponding derivation is:

```
DQ(FQ) =
    CREATE PROCEDURE increase_price (nPrice NUMBER)
    BEGIN
        UPDATE flights SET price = price*1.1
        WHERE
            airline IN {'AL02', 'AL03', 'AL05'} and
            price = &nPrice
    END;
```

3.3 Function

```
DS(
    CREATE OR REPLACE FUNCTION <function_name>
    ([<parameter_name> IN|OUT <datatype>
    [,<parameter_name> IN|OUT <datatype> ... ]])
    RETURN <datatype> IS
        <variables>|<cursors>|<user-defined-exceptions>
    BEGIN
        <SQLf>|<fuzzy_procedural_sentences>
        EXCEPTION <actions>
    END
) =
    CREATE OR REPLACE FUNCTION <function_name>
    ([<parameter_name> IN|OUT <datatype>
    [,<parameter_name> IN|OUT <datatype> ... ]])
    RETURN <datatype>  IS
        <variables>|DS(<cursors>)|DS(<user-defined-exceptions>)
    BEGIN
        DS(<SQLf>)|DS(<fuzzy_procedural_sentences>)
        EXCEPTION DS(<actions>)
    END
```

The user requirement Find destination quantity for those flights whose origin is Caracas, but with any high duration, with a threshold of 0.6, may be expressed as:

```
FQ =
    CREATE FUNCTION flights_quantity() RETURNS INTEGER
```

```
BEGIN
    DECLARE n_TAs INTEGER;
    SELECT COUNT(*)  INTO n_TAs FROM flights AS flightsx
    WHERE flightsx.origin = 'Caracas' AND NOT FOR SOME  flights AS
    flightsy (flightsy.number = flightsx.number AND
    difference(flightsy.arrival_time, flightsy.departure_time) = high)
    WITH CALIBRATION 0.6; RETURN n_TAs;
END;
```

The corresponding derivation is:

```
DQ(FQ) =
    CREATE FUNCTION flights_quantity()  RETURNS INTEGER
    BEGIN
        DECLARE n_TAs INTEGER;
        SELECT COUNT(*)  INTO n_TAs FROM flights AS flightsx
        WHERE flightsx.origin = 'Caracas' AND NOT FOR SOME  flights AS
        flightsy (flightsy.number = flightsx.number AND
        difference(flightsy .arrival_time - flightsy.departure_time) >= 8.5);
        RETURN n_TAs;
    END;
```

Difference is a user defined function that returns number of hours between arrival_time and departure_time attributes.

3.4 Trigger

```
DS(
    CREATE TRIGGER <name> ... WHEN (<fuzzy condition>)
    [WITH CALIBRATION λ]
    {<SQLf>} )
=
    CREATE TRIGGER <name> ...
    WHEN DNC(<fuzzy condition>, ≥,λ)
    DS({<SQLf>})
```

The user requirement Increase prices in 10% of those flights whose airlines are good, with a threshold of 0.6, in case of inserting a new flight, may be expressed as:

```
FQ =
    CREATE TRIGGER BestFlights
    BEFORE INSERT ON flights WHEN (:new.airline = good)
    FOR EACH ROW new.price: = new.price*1.1 WITH CALIBRATION 0.6
```

The corresponding derivation is:

```
DQ(FQ) =
    CREATE TRIGGER BestFlights BEFORE INSERT ON flights
    WHEN (:new.airline IN {'AL02', 'AL03', 'AL05'})
    FOR EACH ROW new.price: = new.price*1.1
```

3.5 Procedural instructions

If Statement

```
DS(
    IF <fuzzy condition> THEN  {<SQLf>} [{ELSEIF <fuzzy condition> THEN
    {<SQLf>}}] [ELSE  {<SQLf>}] [WITH CALIBRATION λ] END IF )
=
```

IF DNC(<fuzzy condition>, ≥,λ) THEN DS({<SQLf>})
[{ELSEIF DNC(<fuzzy condition>, ≥,λ) THEN DS({<SQLf>})}]
[ELSE DS({<SQLf>})] END IF

Case Statement

DS(
SELECT CASE(<attribute>) {WHEN <fuzzy predicate> THEN <action>}
END FROM <table> WHERE < fuzzy condition> [WITH CALIBRATION λ])
=
SELECT CASE(<attribute>) {WHEN DNC(<fuzzy predicate>, ≥, λ) THEN
DS(<action>)} END FROM <table> WHERE DNC(<fuzzy condition>, ≥,λ)

Loop Statement

DS(
LOOP {<SQLf>} END LOOP)
=
LOOP DS({<SQLf>}) END LOOP

While Statement

DS(
WHILE <fuzzy condition> DO {<SQLf>}
[WITH CALIBRATION λ] END WHILE)
=
WHILE DNC(<fuzzy condition>, ≥,λ) DO DS({<SQLf>}) END WHILE

Repeat Statement

DS(
REPEAT {<SQLf>} UNTIL <fuzzy condition>
[WITH CALIBRATION λ] END REPEAT)
=
REPEAT DS({<SQLf>}) UNTIL DNC(<fuzzy condition>, ≥,λ) END REPEAT

For Statement

DS(
FOR result AS <SQLf> DO {<SQLf>} END FOR)
=
FOR result AS DS(<SQLf>) DO DS({<SQLf>}) END FOR

For example, the user requirement *Find flight numbers and approached flight time using good airlines, with a threshold of 0.6*, may be expressed as:

```
FQ =
    SELECT number,
        CASE difference(arrival_time, departure_time)
            WHEN high THEN 'High' BREAK;
            WHEN medium THEN 'Medium'
            WHEN low THEN 'Low' BREAK;
        END
    FROM flights WHERE airline = good
    WITH CALIBRATION 0.6;
```

The derivation principle applied to this statement produces the regular statement:

```
DS(FQ) =
    SELECT number,
        CASE difference(arrival_time, departure_time)
            WHEN ≥ 0 and ≤ 3.5 THEN 'Low' BREAK;
            WHEN ≥ 3.5 and  ≤ 6.5 THEN 'Medium' BREAK;
            WHEN ≥ 8.5 THEN 'High' BREAK;
        END
    FROM flights WHERE airline IN {'AL02', 'AL03', 'AL05'};
```

4. STATEMENTS PROCESSING

The processing mechanism based in the derivation principle consists in the following steps. First, we apply this principle in order to obtain the corresponding derived statements (that are crisp). Second, we evaluate resulting statements using a regular Relational Database Management System. The main idea in this mechanism is take advantage of existing technology into the REDMS. It leads to a low cost of implementation. On the other hand, derivation principle may be used in order to minimize the total spent time in fuzzy sentences.

We have built a fuzzy querying system for SQLf [6] that processes SQLf statement. This system is a layer over Oracle RDBMS. It is composed of the modules:

Client's Interface. It receives user's fuzzy sentences and shows the fuzzy query answer.

Dispatcher. It receives fuzzy sentence and delivers at remainder of the modules the necessary structures for the instruction execution, and to receive the respective answers from them.

Sentence Analyzer. It checks syntactic and semantic correctness of the statements. In case of fuzzy queries, the Analyzer builds a tree structure for the fuzzy query that is used in the evaluation process.

Query's Translator SQLf → SQLf-92 and SQL-99. It is the responsible for translating queries that contain fuzzy terms in regular queries using the Derivation Principle and producing queries in SQL-92 (SQL2) and SQL-99 (SQL3) norms.

Evaluator/Calibrator. It performs the evaluation of fuzzy queries e interacts with the RDBMS in order to retrieve database element relevant to the query processing. The Evaluator/Calibrator computes the satisfaction degrees and calibrates the answer of the fuzzy query. All fuzzy term is stored in a Fuzzy Terms Catalog.

5. CONCLUSION

With the expansion of massive trading and information systems which need active database features and user preference handling, database languages as SQLf3 promise to be very useful. Applications using SQLf3 features can achieve more flexible requirements, offering primordial information to the user that is not represented in a regular query.

SQLf already possesses an evaluation mechanism of queries based on Derivation Principle. Derivation Principle obtains classic queries from fuzzy ones. The fuzzy query is evaluated on the derived regular query result. This mechanism reduces the cost fuzzy queries processing, which is the main problem with the fuzzy databases. We have applied this principle to fuzzy procedural statements: anonymous block, stored procedures, functions, triggers and procedural instructions. This work has presented the transformation schemes for evaluating such statements with the Derivation Principle mechanism.

We have implemented this evaluation mechanism into a flexible querying system prototype called SQLfi [6]. We hope with this prototype to make experimental studies of the evaluation mechanism behavior. We points also to the application of the derivation principle to other features of SQLf3: fuzzy integrity constraints, abstract data types and extended assertions.

The application of the derivation principle presented here was limited to queries that allow strong derivation. It is necessary to apply and study the derivation principle when the fuzzy sentences do not allow strong but weak derivation.

Acknowledgments

This work has been made thanks to the inspiration of a wonderful person in ours lives, who has paid the price for giving us all that we need for success and all that we need for live; moreover, this person has given us the peace and the salvation. We give a great acknowledgment to the King of Kings and Lord of Lords, Jesus Christ.

We would also express grateful to our friend Eng. Juan Carlos Eduardo who has believed in our work and has give us an important collaboration in the implementation of SQLf procedural features into a fuzzy querying system prototype.

References

[1] ANSI X3, "Database Language SQL", 135-1992, American National Standards Institute, New York.

[2] Bosc, P. , Dubois, D., Pivert, O. and Prade, H. "Flexible Queries in Relational Databases - The Example of the Division Operator", TCS 171(1-2): 281-302 (1997).

[3] Bosc, P. and Pivert, O. “On the efficiency of the alpha-cut distribution method to evaluate simple fuzzy relational queries”, Advances in Fuzzy Systems-Applications and Theory, Vol 4, Fuzzy Logic and Soft Computing, B. Bouchon-Meunier, R.R. Yager, L.A. Zadeh eds, World Scientific, pp 251-260, 1995.

[4] Bosc, P. and Pivert, O. “SQLf: A Relational Database Language for Fuzzy Querying”, IEEE Transactions on Fuzzy Systems, Vol 3, No. 1, Feb 1995.

[5] Bosc, P. and Pivert, O. “SQLf Query Functionality on Top of a Regular Relational Database Management System”, Knowledge Management in Fuzzy Databases, Pons, O., Vila, M. and J. Kacprzyk (Eds.), Physica-Verlag, (2000), Pp. 171-190.

[6] Eduardo, J., Goncalves, M. and Tineo, L. “A Fuzzy Querying System based on SQLf2 and SQLf3”, The XXX Latin-American Conference on Informatics, Sep 2004.

[7] Goncalves, M. “Extensión del Lenguaje de Interrogación Flexible a Bases de Datos SQLf mediante las normas SQL2 y SQL3”. Informe final de Trabajo Especial de Grado presentado ante U.S.B., Septiembre 2001.

[8] Goncalves, M. and Tineo, L. “SQLf Flexible Querying Language Extension by means of the norm SQL2”, The 10th IEEE International Conference on Fuzzy Systems, Vol 1, Dec 2001.

[9] Goncalves, M. and Tineo, L. “SQLf3: An extension of SQLf with SQL3 features”, The 10th IEEE International Conference on Fuzzy Systems, Vol 3, Dec 2001.

[10] Goncalves, M. and Tineo, L. “WWW Data Source Selection with SQLfi”, FUZZ-IEEE 2005, May 2005.

[11] López, Y. and Tineo, L. “About the Performance of SQLf Evaluation Mechanisms” , The XXX Latin-American Conference on Informatics, Sep 2004.

[12] Melton, J.. “ISO/ANSI Working Draft: Database Language SQL (SQL3)”, X3H2-93-091/ISO DBL YOK-003.

[13] Tineo, L. "Interrogaciones Flexibles en Base de Datos Relacionales", Trabajo de Ascenso presentado ante U.S.B., Enero 1998.

[14] Tineo, L. “Extending RDBMS for Allowing Fuzzy Quantified Queries”, Lecture Notes in Computer Science, Vol. 1873, Mohamed Ibrahim-Josef Küng-Norman Revell (Eds.) Springer Verlag, (2000), Pp. 407-416.

SQLf Horizontal Fuzzy Quantified Query Processing

Leonid Tineo
Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
leonid@usb.ve

Abstract

In order to make more flexible database access the query language SQLf has been previously proposed. This language allows the expression of user preferences in queries giving discriminated answers. Such preferences are specified by means of fuzzy logic conditions. User may also specify the quality of the desired answer giving a minim satisfaction level. One of the SQLf features is the use of Fuzzy Quantifiers in the WHERE clause as an operator for criteria combination. This use is known as Horizontal Quantification. An open issue for flexible querying is the problem of providing efficient evaluation mechanisms. As ever, these queries may be processed with a Naïve Strategy processing the fuzzy quantified sentence over the e base relation obtained by the FROM clause. We propose here two improved strategies based in the application of the Derivation Principle. This principle consists in the distribution of a minimum user desired satisfaction degree over query involved conditions. One mechanism uses an external program in order to process the fuzzy query over the result of a SQL query. The other one performs the fuzzy query processing via a function calls decorated SQL query. Such functions perform the computation fuzzy sentences satisfaction degrees. We present in this paper a formal performance study of these three mechanisms. This study has been made using a SQLf prototype build on top of a RDBMS.

Keywords: Query Processing, Fuzzy Querying, Querying Performance, Fuzzy Quantifiers, SQLf.

1. INTRODUCTION

SQLf provides a large variety of fuzzy querying constructions, allowing the use of fuzzy conditions in any place where SQL allows a Boolean condition [3]. An interesting feature of SQLf is allowing the use of fuzzy quantifiers in querying. Liétard [5] and Tineo [8] have make contributions in the semantics of fuzzy quantified sentences in database querying.

Nevertheless, the problem of database fuzzy querying not is only a semantics issue but also a practical reality. The interest of some previous works has been to provide efficient evaluation mechanisms for fuzzy querying language SQLf [1], [3], [5]. At present time, evaluation mechanisms for fuzzy quantified queries have been proposed and studied only for vertical quantification where the quantification is made over the number of rows satisfying a fuzzy condition. In this context: Bosc et al [1] have presented a strategy based in Sugeno Integral properties for improve query evaluation, we name it Sugeno Strategy; Tineo [9] has presented the application of the Derivation Principle to such queries. This principle consists in distributing the minim satisfaction degree of desired solution deriving form the fuzzy query a crisp. The fuzzy query is then evaluated over the result of the regular derived query. This strategy has been proposed to be used several querying SQLf structures [4], [9]. López and Tineo [6] have made a formal statistics study of the performance some vertical fuzzy quantified queries evaluation mechanisms: Sugeno, Derivation and Naïve.

We are interested here in queries with horizontal quantification. These queries, given fuzzy criteria list, quantify the number of criteria that are satisfied for each row. We present here evaluation mechanisms for horizontal quantified query processing. For such mechanisms, we make a system performance analysis that is based in experimentation and use of statistics models [7]. The analysis of such experiments will lead us to distinguish the effects of factors that may inside in the performance of query evaluation mechanisms, pointing to the selection of the evaluation mechanisms with best performance.

Fuzzy quantifiers may be classified according to their nature in absolute and proportional. According to the behaviour of their membership functions, fuzzy quantifiers may be classified in increasing, decreasing and unimodal ones [11]. We restrict our work to increasing absolute fuzzy quantifiers. This restriction is made with loose of generality, because for the kind of queries studied here, the use of different kinds of quantifiers are equivalent [10].

The rest of this paper is organized as follows: In section 2 we present the syntax and semantic of the SQLf querying structure matter of this work, we give there an illustrative example that will be used in further sections of this paper. Section 3 is devoted to present the evaluations mechanisms that have been proposed for SQLf query evaluation and our new proposal for applying them in the evaluation of horizontal quantified queries. In order to study the performance of these evaluation mechanisms, in section 4 we set down our experiments. Results obtained form these experiments are shown and analyzed in section 5. Finally, in section 6 we present conclusions that we have arrived form this work, in that section, we also point to further works.

2. QUERY'S SEMANTICS

SQlf provides a kind of horizontal quantified queries which general form is:

select *A* from *R* where *Q(fc1,..,fcn)* with calibration *t*.

Being *t* a threshold of the minim satisfaction degree for user desired rows in the answer, *A* an attribute (attributes list) of the relation (relations list) *R*, *Q* a fuzzy quantifier, and *fc1,..,fcn* a list of fuzzy conditions. This query returns the fuzzy relation *Rf* on $\{a / (\exists y \in R / y.A=a) \wedge (\mu(Q(X,fy)) \geq t) \}$, being the membership degree of each element *a:* $\mu Rf(a) = \mu(Q(X,fy))$ (the truth degree of fuzzy quantified sentence *Q X's are fy*), where $X=\{fc1,..,fcn\}$, *fy* is a fuzzy predicate on *X* whose satisfaction degree is $\mu fy(fc)=\mu fc(y)$ (the satisfaction degree of the row *y* to the fuzzy condition *fc*). The sentence *Q X's are fy* is interpreted with the Yager's decomposition interpretation [11]. This interpretation coincides with Tineo's [8] one in this case. The satisfaction degree of the quantified sentence is $\mu(Q(X, fy)) = \sup(\min(\mu Q(i), \mu fyi))$ where μfyi is the i-th higher value of the $\mu fy(fc)$ degrees.

For example, let's consider the employee relation of **Table 1**. We search for employees that meet at least three of the criteria: be around 34 years old, tall, heavy, of a high level study and with a regular salary. Fuzzy terms involved in this requirement may be defied by fuzzy sets in **Fig. 1**. We ask only for answers with satisfaction degree greater or equal to 0.5. This query is expressed in SQLf as:

```
select * from employee where
      at_least_3(
                  age=aprox_34_years, height=tall, weight=heavy, studies=high_level,
                  salary=regular_salary
      )
with calibration 0.5.
```

Table 1. Employee Relation

id_num	first_name	last_name	age	Height	Weight	studies	salary	Profession
Emp01	Ann	Tao	40	168	58	4	1250	Physician
Emp02	Cri	Sto	33	200	77	5	1500	Engineer
Emp03	Kal	Hil	33	183	92	1	500	Watchman
Emp04	Ken	Cha	36	171	90	4	1250	Architect
Emp05	Leo	Tin	37	180	80	2	850	Technician
Emp06	Liz	Yun	38	155	61	3	1000	Technician
Emp07	One	Azo	32	165	65	2	750	Secretary
Emp08	Rod	Bin	34	192	120	1	500	Watchman
Emp09	Tez	Mum	35	175	110	4	1000	Nurse
Emp10	Zul	Gaz	42	170	65	2	750	Secretary

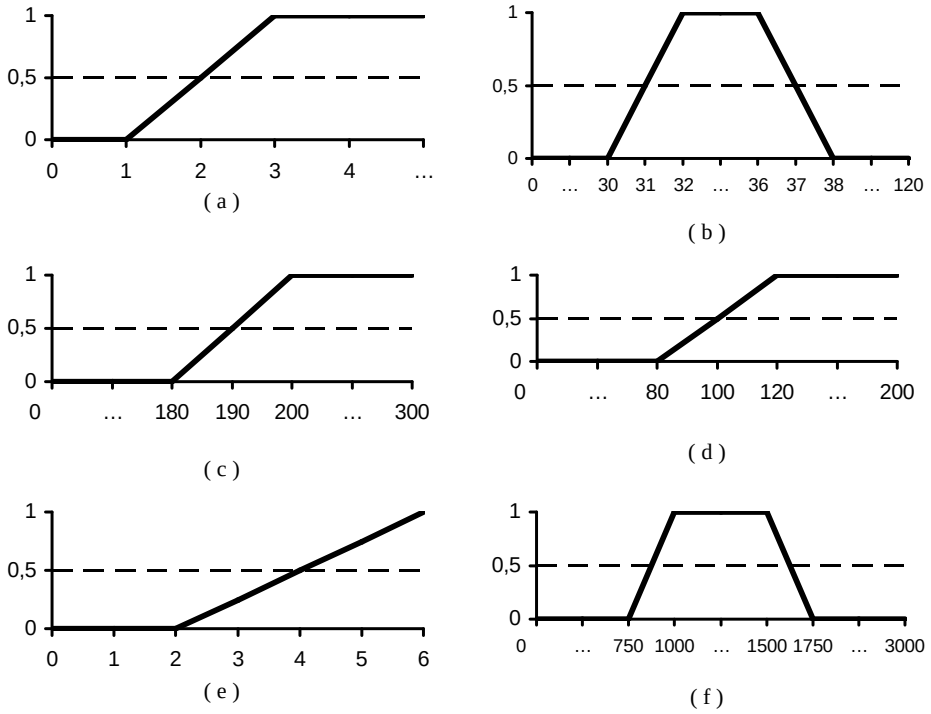


Fig. 1. Membership functions of fuzzy sets defining fuzzy terms: (a) at_lest_3. (b) aprox_34_years. (c) tall. (d) heavy. (e) high_level. (f) regular_salary.

In order to show the semantics of the example query, we present here some tables with the computation of the query result:

First, for each row in employee relation (**Table 1**), we show (**Table 2**) the satisfaction degree of fuzzy predicates defined in **Fig. 1**.

Applying the Yager's decomposition interpretation [11] for quantified propositions, the computation of the satisfaction degree for the fuzzy quantified sentence is made for each row of the employee relation, sorting in decreasing order the satisfaction degree of the fuzzy criteria (**Table 2**) and mixing them with the satisfaction degrees of the fuzzy quantifier according to its definition given in **Fig. 1**. This computation of satisfaction degrees for fuzzy quantified sentences is shown in **Table 3**.

Finally, the query returns the fuzzy relation in **Table 4**. It contains those rows which satisfaction degree is greater or equal to the minimum desired satisfaction level, in this case 0.5. The satisfaction degree of each row is part of the result. Rows are decreasing ordered by this degree.

Table 2. Satisfaction Degree of Fuzzy Criteria

y.id_num	*µfy(age= aprox_34_years)*	*µfy(height= tall)*	*µfy(weight= heavy)*	*µfy(studies= high_level)*	*µfy(salary= regular_salary)*
Emp01	0	0	0	0,5	1
Emp02	1	1	0	0,75	1
Emp03	1	0,15	0,3	0	0
Emp04	1	0	0,25	0,5	1
Emp05	0,5	0	0	0	0,4
Emp06	0	0	0	0,25	1
Emp07	1	0	0	0	0
Emp08	1	0,6	1	0	0
Emp09	1	0	0,75	0,5	1
Emp10	0	0	0	0	0

Table 3. Satisfaction Degree of Horizontal Fuzzy Quantified Sentence

y.id_num	min(*µQ(1) ,µfy1)*	min(*µQ(2) ,µfy2)*	Min(*µQ(3),µfy3)*	min(*µQ(4,µfy4)*	min(*µQ(5),µfy5)*	*µ(Q(X,fy))*
Emp01	0	0,5	0	0	0	0,5
Emp02	0	0,5	1	0,75	0	1
Emp03	0	0,3	0,15	0	0	0,3
Emp04	0	0,5	0,5	0,25	0	0,5
Emp05	0	0,4	0	0	0	0,4
Emp06	0	0,25	0	0	0	0,25
Emp07	0	0	0	0	0	0
Emp08	0	0,5	0,6	0	0	0,6
Emp09	0	0,5	0,75	0,5	0	0,75
Emp10	0	0	0	0	0	0

Table 4. Fuzzy Relation resulting form the Fuzzy Query

Id_num	first_name	last_name	Age	height	Weight	studies	salary	Profession	*µRf*
Emp02	Cri	Sto	33	200	77	5	1500	Engineer	1
Emp09	Tez	Mum	35	175	110	4	1000	Nurse	0,75
Emp08	Rod	Bin	34	192	120	1	500	Watchman	0,6
Emp01	Ann	Tao	40	168	58	4	1250	Physician	0,5
Emp04	Ken	Cha	36	171	90	4	1250	Architect	0,5

3. EVALUATION MECHANISMS

Proposed SQLf processing mechanisms are intended for adding fuzzy querying capabilities on top of an existing RDBMS. Such mechanisms perform fuzzy query evaluation on the result of an underlying regular query addressed to the RDBMS. In this context a theoretical measure of mechanisms behaviour is the number of accessed database rows. One may think that whenever more rows are accessed more time is spent. Evaluation mechanisms have been proposed in order to keep low the number of accessed rows. It will be interesting to show the application of such mechanisms in case of horizontal fuzzy quantified queries. On the other hand, in some cases, regular selection criteria may be of high complexity and it may have a bad incidence in performance. Therefore it will be helpful to make performance study.

Naïve Strategy *[1], [3]* consists in a program scanning the whole database relation and computing the satisfaction degrees. For the example in previous section, Naïve strategy will access all rows in **Table 1**, computing all fuzzy criteria degrees in **Table 2** and all the fuzzy quantified sentence satisfaction degrees in **Table 3**. Each row must be retained if pass the desired threshold. In the example, this mechanism accesses the double quantity of rows that actually are in the solution. For each one computes fuzzy sentence satisfaction degrees which is also time consuming. The pseudo code for this evaluation strategy applied to horizontal quantified queries' processing is:

```
Procedure Naïve
Begin
    Result ← ∅
    For y in (select * from R) do
        M ← SatisfactionDegree(y,Q(fc1,..,fcn))
        If M ≥ t then
            Result ← Result ∪ {<y.A,M as mu>}
    select * from Result order by mu desc
End
```

The pseudo code of the function for computing fuzzy quantified sentence satisfaction degree for each row is:

```
Function SatisfactionDegree(y,Q(fc1,..,fcn))
Begin
    Compute µfy(fci), for i∈{1,..,n}
    Sort µfy(fc1),..,µfy(fcn) in decreasing order
    Let's <µfy1,..µfyn> be the resulting sort
    Return sup min(µQ(i),µfyi), i∈{1,..,n}
End
```

Sugeno Strategy *[1]* consists in scanning the whole table as in Naïve one, but with halting conditions for each group of elements under the quantification. The idea is avoid access to some elements. Nevertheless, as in horizontal quantification, elements under quantification are not rows but fuzzy criteria, these heuristics lose their sense since each row must actually be accessed. We consider by this reason that this strategy have no benefit for this kind of queries. Therefore, it will not be considered in the following of this work.

Derivation Principle *[9]* consists in processing rows retrieved by a regular query intended for selecting rows whose satisfaction degree is greater or equal to the given threshold *t*. This set of rows is known in fuzzy sets theory as the *t*-cut [2]. Given the fuzzy quantified query, we derive a regular query intended for retrieving the *t*-cut of the fuzzy query answer set. Some times, it is impossible to derive such query due to fuzzy condition nature, but a query for a superset of the *t*-cut. In this case we say that the derivation is weak otherwise is said to be strong.

For a SQLf horizontal quantified query of form:

select *A* from *R* where *Q(fc1,..,fcn)* with calibration *t*.

we obtain the derived classical SQL query:

select * from *R* where OAC({DNC(*fc1,≥,t*),..,DNC(*fcn,≥,t*)},*LB*).

Being OAC({DNC(*fc1,≥,t*), .., DNC(*fcn,≥,t*)}, *LB*) the normal disjunctive from containing as terms all conjunction combinations of *LB* elements from {DNC(*fc1,≥,t*), .., DNC(*fcn,≥,t*)}, *LB* is the lowest natural which satisfaction degree to the quantifier is greater or equal to *t*, DNC(*fci,≥,t*) is the derived necessary condition for the *t*-cut of *fci*. Definition of DNC(*fci,≥,t*) has been matter of previous works [2], [4]. If all DNC(*fci,≥,t*) are strong derivation, then the derived query presented here is also a strong derivation. We have made formal proof of this query derivation that will appear in [10]. Proof is hard and very long so it could not be included in this paper. Interested reader might contact the author.

For our example query presented in previous section, with the definition of fuzzy terms given in **Fig. 1**, we obtain the derived query:

```
select * from employee where
  (31<=age and age<=37 and 190<=height) or
  (31<=age and age<=37 and 100<=weight) or
  (31<=age and age<=37 and 4<=studies) or
  (31<=age and age<=37 and 875<=salary and salary<=1625)or
  (190<=height and 100<=weight) or
  (190<=height and 4<=studies) or
  (190<=height and 875<=salary and salary<=1625) or
  (100<=weight and 4<=studies) or
  (100<=weight and 875<=salary and salary<=1625) or
  (4<=studies and 875<=salary and salary<=1625).
```

With the Derivation Principle in this example, just rows from the employee relation (**Table 1**) that are in the final result (**Table 4**) must be accessed in this fuzzy query processing. This shows that Derivation Principle keeps low the number of rows accessed in query processing.

The Derivation Principle may be used into two query evaluation mechanisms: First is the Program Derivation Strategy consists in processing the fuzzy query over the result of derived query by means of an external program.

```
Procedure Derived Program
Begin
    Result ← ∅
    For y in (select * from R where OAC({DNC(fc1,≥,t),..,DNC(fcn,≥,t)},LB)) do
        M ← SatisfactionDegree(y,Q(fc1,..,fcn))
        Result ← Result ∪ {<y.A,M as mu>}
    select * from Result order by mu desc
End
```

Second is the Query Derivation Strategy consists in adding to the "select" clause a call to a function that computes the satisfaction degree of the fuzzy sentence. Also an "order by" clause is added for giving the result in decreasing order of the computed satisfaction degrees. We may appreciate this mechanism in the pseudo code:

```
select A, SatisfactionDegree(y,Q(fc1,..,fcn)) as mu from R y
where OAC({DNC(fc1,≥,t),..,DNC(fcn,≥,t)},LB) order by mu desc
```

Despite the Derivation Strategy requires the computation of satisfaction degrees for a low number of rows than Naïve Strategy, we must to show experimentally if it really gives better performance. It may be natural to think that the use of a rather complex condition in the "were" clause of derived query might involve high cost giving advantage to the Naïve strategy. On the other hand it is necessary also to consider two proposed mechanisms based in the Derivation Principle because some DMBS do not allow function calls in querying structure.

4. EXPERIMENTS' DESIGN

The performance evaluation will be made using formal model statistic method. The idea of this method is to explain the influence of several considered factors in the observed values from experiments. The importance of a factor is measured by the proportion of the total variation in the response that is explained by the factor.

The queries where addressed to the database relation: employee(id_number, first_name, last_name, age, height, weight, studies, salary, profession, emp_dep_num). Database relations are populated using a program in PL/SQL that generates database extension. Values for id_number are sequential generated. Attributes first_name, last_name and profession are single function of id_number. Foreign key emp_dep_num values are random generated into a range. Rest of employee relation's attributes are uniform random generated. Attribute age ranks between 18 and 65 years old. Attribute height ranks between 120 and 210 centimeters tall. Attribute weight ranks between 50 and 150 kilograms. Attribute studies ranks between 1st and 6th academic level. Attribute salary ranks between 100 and 3000$.

The variables that are observed in the experiments are called answer variables. They usually are measures of the system behavior. In our case we are interested in the performance of the fuzzy query answer system. Therefore we want to observe the total spent time in solving the fuzzy query.

The experimental factors are those parameters whose values are changed in the experiments in order to determine their effect in the answer variable. First considered factor is the strategy used. We expect this factor to have a high influence in the performance of the query evaluation. The levels for this factor are: Naïve, Derivation (Program) and Query (Derived), respectively. The volume of data is a factor that must be considered. The access time is always depending of the volume of data. Despite in the different strategies, the proportion of accessed registers does not depends on how large is the database, it is reasonable to think that an interaction might exists between the volume factor and the strategy factor. We define six levels for the volume factor, varying the quantity of rows in the *employee* table of the database: Very (400 rows), Tiny (800 rows), Small (1600 rows), Medium (3500 rows), Large (7000 rows) and Extra (14 rows). The selectivity of a query may influence its performance. A query is more selective in the measure that retrieves a smaller set of result. We may vary the selectivity of a SQLf query changing its desired satisfaction degree level t. We have chosen three levels for selectivity factor: Low (t = 0.25), Middle (t = 0.5) and High (t= 0.75).

A condition that could vary in a database experiment is the physical storage structure for the database tables. We prefer in this study to avoid this kind of consideration. We adopt to have simple table structures provided of an index for each attribute in the relation. We must remember that evaluation mechanisms proposed and analyzed in this work are designed as a logic layer on top of an existing RDBMS.

We will fix conditions and fuzzy quantifier in the query. The design of fuzzy terms involved is made with care to be really representative. As data is generated with uniform distributions, definitions of such fuzzy terms are made in a way of be in some sense "uniform". Fuzzy terms definitions in **Fig. 1** have this desired characteristic. For the experimentation, we use the same query presented as example here before, varying the value of t in the "with calibration" clause:

```
select * from employee where
    at_least_3( age=aprox_34_years, height=tall, weight=heavy,
                studies=high_level, salary=regular_salary
    )
with calibration t.
```

The repetitions of experiments are called replicas. As we use a dedicated server, replicas are unnecessary. Nevertheless, we perform three replicas of each experiment in order to prove this hypothesis. Replicas will no be considered in the model.

We have chosen a full factorial design for our experimental study. That is, we will consider all the mentioned factors and all their levels. We have the hypothesis that all factors and their interactions have significant influence in the performance. This kind of design allows study the influence of each factor and all theirs interactions. We must take value for the answer variable for each possible combination of the factors in all their levels.

The model is an expression of the observed values for the response variable as a combination of experimental factors levels influences. The model for our study is: $y_{ijk} = \mu_{...} + (E_i + V_j + T_k) + (EV_{ij} + ET_{ik} + VT_{jk}) + EVT_{ijk})$. Being: y_{ijk} the observed value for the levels *i, j, k* of the factors *E, V* and *T*, respectively; $y_{...}$ the arithmetic mean of the observed values of all experiments; F_m the effect of the factor F at the level m, $F \in \{E,V,T\}$; $F F_{m'm''}$ the effect of the Interaction between factors F' and F'' at the levels m' and m'', respectively, with different $F', F'' \in \{E,V,T\}$; and EVT_{ijk} the effect of the Interaction between all the factors *E*, *V* and *T* for the levels *i, j* and *k*, respectively.

5. EXPERIMENTAL RESULTS

We have run the experiments in a SQLf prototype on top of Oracle 9i RDBMS. Experimentation platform was a desktop computer with 895MHz Pentium III processor, 512MB RAM and 20GB hard disk, running Red Hat Linux 8.0, Query transformation mechanisms where coded in SWI Prolog.

Experiments have given the results summarized in **Table 5**. We have loaded these results in the R statistical software with the experimental model. **Table 6** shows the analysis of variance for this model with the observer data.

Table 5. Observed Times. We show here the mean of the three replicas

	Naïve			**Derivation**			**Query**		
	Low	**middle**	**high**	**low**	**middle**	**High**	**Low**	**middle**	**high**
Very	2,24	2,02	1,87	2,12	2,03	1,70	1,24	1,05	0,90
Tiny	2,61	2,56	1,98	2,52	2,14	1,74	1,31	1,14	0,95
Small	3,47	3,00	2,35	3,27	3,16	2,18	1,76	1,55	1,07
medium	5,30	4,39	2,95	5,12	4,59	2,02	2,69	2,30	1,13
Large	8,89	8,39	3,97	8,80	7,66	2,69	4,67	4,63	1,47
Extra	15,57	12,78	6,24	15,21	12,30	3,64	8,54	6,65	1,93

Table 6. ANOVA of Full Factorial Model. All factors and thir interactior are very significant

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
Volume	5	1151.09	230.22	2312.2050	< 2.2e-16	***
Selectivity	2	270.70	135.35	1359.3842	< 2.2e-16	***
Strategy	2	198.86	99.43	998.6075	< 2.2e-16	***
Volume:Selectivity	10	287.12	28.71	288.3684	< 2.2e-16	***
Volume:Strategy	10	84.74	8.47	85.1083	< 2.2e-16	***
Selectivity:Strategy	4	14.85	3.71	37.2855	< 2.2e-16	***
Volume:Selectivity:Strategy	20	15.60	0.78	7.8335	2.263e-13	***
Residuals	108	10.75	0.10			

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1						

The summary of the analysis of variance (ANOVA) obtained for the full factorial model with experimental data is presented in **Table 6**. It shows that the stochastic model selected fits very well to the data. This fact tells us that we may take conclusions about the performance using this experiment, because it is stochastically valid. We can see also in the ANOVA table that all interactions between factors are very significant to explain the results. Therefore, we will plot and analyze all interactions of two factors. We have confirmed the hypothesis presented in the experiment design about the significance of selected factors and their interactions.

Let's analyze the interaction between Volume and Strategy factors, shown in **Fig. 2**: As we may expect, processing times increases with increasing of database volume. Times for the Query Derivation strategy are considerably lower than others. On the other hand, times for the Naïve and the Derivation Program strategies are very similar being the last lightly lower than the first.

Lowest times for Query Derivation strategy obeys to the fact that this strategy gives the whole control to the DBMS, allowing the computation of satisfaction degrees for each row while they are selected. In this case, due to the application of the Derivation Principle, satisfaction degrees are calculated only for those rows that meet the threshold specified in the calibration. In the other cases the set of rows must be retrieved from the database and after a program must scan this set of rows in order to compute satisfaction degrees. This kind of "re-scan" explains higher costs.

Difference between Derived Program and Naïve strategies is due to the fact that Derived Program works with a subset of rows selected by the derived query while the Naïve strategy works with all rows from the base relation. Naïve strategy involves extra computing due to calculation of satisfaction degrees of rows that will be rejected due to the calibration. On the other hand, in this case, due to the application of the Derivation Principle, Derived Program strategy only computes satisfaction degrees just for those rows that meet the threshold specified in the calibration.

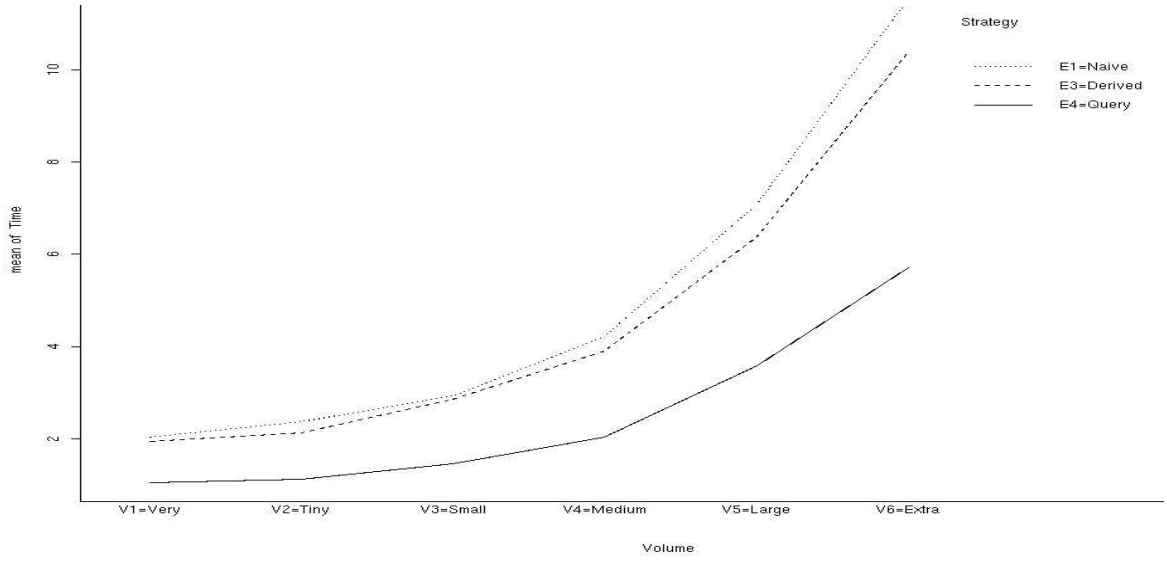


Fig. 2. Volume-Strategy Interaction

Let's analyze the interaction between Selectivity and Strategy factors (**Fig. 3**): Query processing time decreases while selectivity increases because a more selective query retrieves a smaller set of answers.

In case of low selectivity we may observe that times for Derived Program strategy and Naïve strategy are very close. It is due to the fact that a low selectivity implies that the set of selected answers is closer to the set of all rows. In this case we have a low number of rows for which the Naïve strategy computes the satisfaction degrees but are then rejected due to the calibration.

In case of middle selectivity, times observed for Derived Program and Naïve strategies are also close. Difference increases just a little. The number of rejected rows for which Naïve strategy makes non useful computation is a bit greater than previous case.

In case of high selectivity, we have a really marked difference between times observed for Derived Program and Naïve Strategies. Here there is high number of rows for which Naïve strategy computes the fuzzy quantified sentence satisfaction degree for being after rejected due to degrees lower than the minim desired degree specified in the calibration of the query. Here times for Derivation Program strategy are going down fast. These times become closer to the Derived Query strategy. It is because the number of desired answers has been reduced with high selectivity, therefore there is a reduced quantity of rows that Derivation Program strategy must re-scan for satisfaction degrees computation.

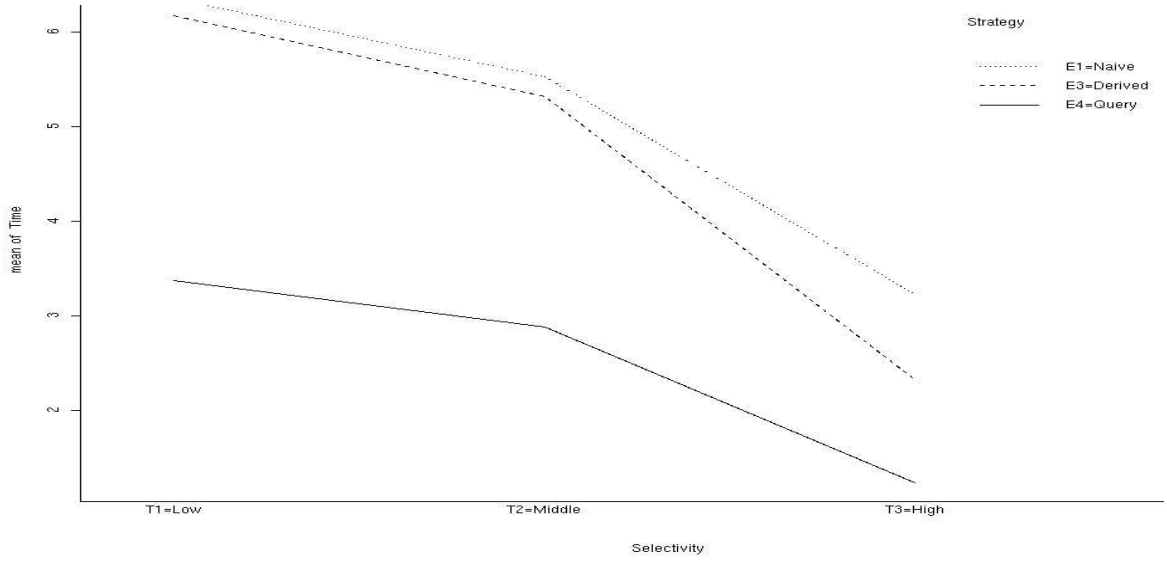


Fig. 3. Selectivity-Strategy Interaction

Let's analyze the interaction between Volume and Selectivity factors, shown in **Fig. 4**: As ever, times grow with volume and are lower as selectivity is higher. Curve for high selectivity is softer that curves for low and middle selectivity. It obeys to the fact that high selectivity keeps low the number of retrieved rows. Times for low selectivity and middle selectivity are very close. To explain that we must remember that selectivity in the experiment has been set bay means of the minim desired satisfaction degree specified in the calibration of the query. Values selected for this parameter where 0.25 (low selectivity) 0.5 (middle selectivity) and 0.75 (high selectivity). Although 0.5 is in the middle between 0.25 and 0.75, the quantity of retrieved rows for the threshold 0.5 is not just in the middle between the cardinalities of retrieved rows sets for 0.25 and 0.75 thresholds.

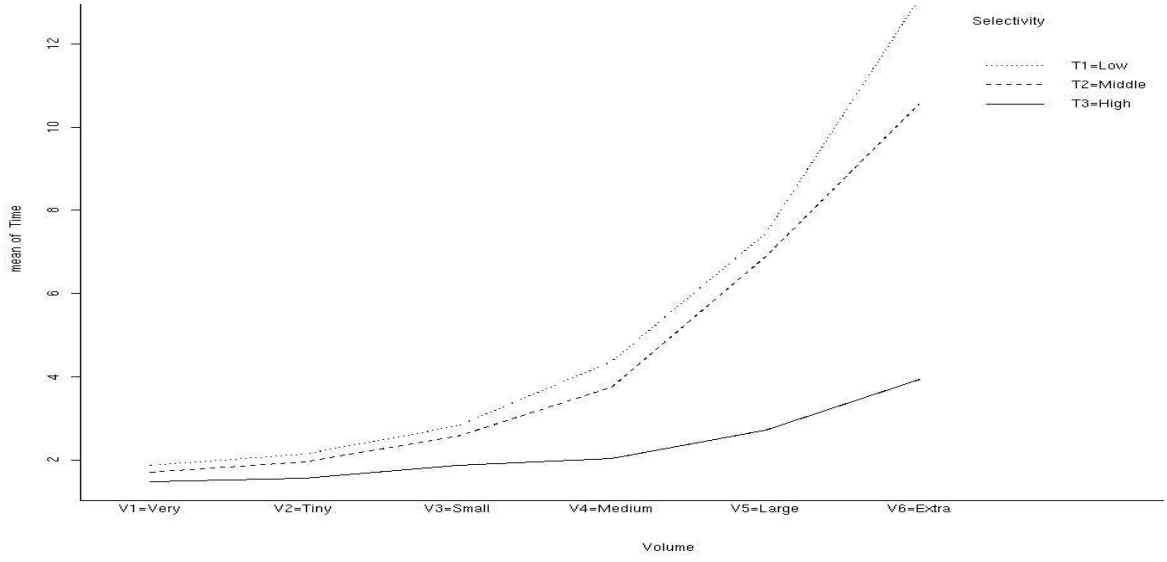


Fig. 4. Volume-Selectivity Interaction

According to these results, we conclude that the Query Derivation strategy is the most suitable mechanism for this kind of queries. Interaction between strategy and volume factors (**Fig. 2**) and interaction between strategy and selectivity factors (**Fig. 3**) clearly show it. Nevertheless, Query Derivation strategy could not be supported in some DBMS because they not allow user defined function calls in queries. In those cases, we Derivation Program strategy would be the most accurate evaluation mechanism because it have better performance than Naïve strategy.

6. CONCLUDING REMARKS

We have dealt here with SQLf horizontal fuzzy quantified queries. We have shown their intuitive and formal semantics. For this kind of queries have discussed about the applicability of known fuzzy query evaluation strategies: Naïve, Sugeno and Derivation Principle. We concluded that Sugeno strategy is not suitable for them. We have introduced the derived query obtained from a horizontal fuzzy quantified query. We have shown with an illustrative example the reduction of row access gained with the Derivation Principle. We have characterized two strategies for query evaluation based in the Derivation Principle: The Derivation Program Strategy and the Derived Query Strategy. We have made a formal statistic study of performance of query evaluation mechanisms presented here. This study involved also the volume and selectivity factors. We have shown with this study that the Derived Query strategy has the best performance between considered mechanisms. It improves considerably the response time respects the Naïve strategy. We have restricted our work to increasing absolute fuzzy quantifiers. Nevertheless, for the kind of queries studied here, the use of different kinds of quantifiers is equivalent. A variant of horizontal fuzzy quantified queries considered here is allowing the fuzzy criteria under quantification to have also preference levels. The study if this kind of queries will be matter of further works. We would also study in further works the application of evaluation strategies presented here in order to build processing mechanisms for other uses of fuzzy quantification in querying such as nesting and division operators. The final objective of our research work would be to implement these query capabilities into a real DBMS.

Acknowledgments

I would like to express a great acknowledgement to the person who has inspirited my work and my life, a very special person who is with me at every moment, giving me love and force for living and guide for doing all things that I do. I have never done this work without the invaluable help that this person has given me. This person is the Lord and Savior of my soul: Jesus Christ, my God.

I also give thanks to my friend Dr. Olivier Pivert who has propose me the initial idea of doing this research and has help with comments about my work for improving it. I also express grateful to my friend Dr. Mariela Curiel who has introduced me to the art of computer systems performance analysis. I beg God to bless theirs lives.

References

[1] Bosc P., Liétard L., Pivert O., "Evaluation of Flexible Queries: The Quantified Statement Case", Proceedings of the 8th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU'2000, Madrid, España, 1115-1122, 2000 .

[2] Bosc, P., Pivert, O., *On the Efficiency of the Alpha-cut Distribution Method to Evaluate Simple Fuzzy Relational Queries*, Advances in Fuzzy Systems-Applications and Theory, Vol. 4, Fuzzy Logic and Soft Computing, Bouchon-Meunier, B., Yager, R. and Zadeh, L. (Eds.), World Scientific, Pp. 251-260, 1995.

[3] Bosc, P., Pivert, O., *SQLf: A Relational Database Language for Fuzzy Querying*, IEEE Transactions on Fuzzy Systems, Vol. 3, N° 1, 1-17, 1995.

[4] Bosc P., Pivert O., "SQLf query functionality on top of a regular relational DBMS", in: Knowledge Management in Fuzzy Databases, O. Pons, M.A. Vila, and J. Kacprzyk (Eds.), Heidelberg: Physica-Verlag, 171-190, 2000.

[5] Liétard L., "Contribution à l' interrogation Flexible de Bases de Données: étude des propositions quantifiées floues" Thèse de Docteur Université de Rennes, 1995.

[6] López Y., Tineo L., "About the Performance of SQLf Evaluation Mechanisms" , Memorias de la XXX Conferencia Latinoamericana de Informática CLEI 2004, Proceeding on CD, 2004.

[7] Raj J., "The Art of Computer Systems Performance", John Wiley/Sons, Inc, 1991.

[8] Tineo L., "A Fuzzy Quantifiers' Interpretation for Database Querying", Proceedings of the First International Conference on Fuzzy Information Processing Theories and Applications FIP 2003. Vol. 1. pp. 423 – 428, 2003.

[9] Tineo L., "Extending the power of RDBMS for Allowing Fuzzy Quantified Queries". Lecture Notes in Computer Sciences, September 2000. Vol. 1873, pp 407-416, 2000.

[10] Tineo L., "A Contribution to Database Flexible Querying: Fuzzy Quantified Queries Evaluation", Doctoral Thesis, Universidad Simón Bolívar, 2005 (to appear).

[11] Yager, R., Interpreting Linguistically Quantified Propositions, International Journal of Intelligent Systems, Vol. 9, (1994), Pp. 541-569.

Fuzzy Object-Oriented Database Language

Julio Luna
Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
99-80696@ldc.usb.ve

and

Leonid Tineo
Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
leonid@usb.ve

Abstract

Formal specification languages are useful tools for unambiguous system specification and implementation correctness proof. Some of these languages incorporate object orientation, which is a powerful tool for system design; this is the case of Object-Z specification language. On the other hand, object orientation has also been used as a data model for database and has been incorporated to DMBS. Nevertheless, these tools fail in modelling real world situations pervaded of fuzziness. Some efforts have been made for incorporating fuzziness to object model, but there is neither a standard nor a formal specification language. In this work, we analyze the object data model and previous fuzzy logic based extensions. At the diverse levels of this model: attribute, object and class level, we propose a way of applying fuzzy logic where needed. We extend Object-Z in order to support fuzzy object oriented features. Though, we bring a powerful tool for specification of database and applications where fuzziness and object orientation are both needed.

Keywords: Formal Specification Language, Fuzzy Object Model, Fuzzy Database.

1. INTRODUCTION

Modeling is a useful tool in system and data engineering. Models may be used to express our understanding of a system, or to describe a product to build. They are also a tool for communication and validation with client and development team. Modeling techniques are often based on diagrams supported by explanatory text. These approaches lack the well-defined and formal syntax that permits precise analysis, and models expressed in these terms are open to possible ambiguity and misunderstanding [12]. At some point we need to express the model more formally and to subject it to rigorous analysis. The use of a formal specification language as a modeling tool provides a way to achieve this. Formal methods are a set of tools that allow the development of a complete, precise and correct specification for system properties and behavior. One commonly used formal specification language is Z, a powerful analytical tool that facilitates system understanding through the development of unambiguous, verifiable mathematical models. Object-Z is an extension of Z that includes object orientation in formal specification [6][7][8]. Nevertheless, there are some problem domains that may not naturally be modeled in precise (crisp) terms, problems 'whose behavior is strongly influenced by human judgment, perceptions or emotions' [13]. Fuzzy sets are intended for treatment of such problems.

On the other hand, databases are fundamental in storage and information retrieval, due to facility, effectiveness, efficiency and security in data handling. As result of birth and success of object orientation, databases passed through adaptations to absorb benefits granted by this model [1]. At present time, 'the advantages of object-oriented databases are acknowledged outside the research and academic worlds and a breakthrough of new commercial softwares is observed' [4].

Current advances in computing open new opportunities and challenges for databases in many emerging applications in fields as multimedia, geographic information systems, knowledge management, CAD/CAM, etc. The object model is suitable for these applications that require the modeling and manipulation of complex objects and semantic relationships. Classical relational database model and its extension of fuzziness do not satisfy the need of modeling complex objects with imprecision and uncertainty. Therefore there is a research tendency on fuzzy object oriented database models in order to deal with complex objects and uncertain [4][9].

We provide here a formal specification tool for object oriented databases allowing modeling of real world situations pervaded fuzziness. This tool is f-Object-Z, a fuzzy extension of Object-Z. It let us have the benefit of a formal modeling tool for databases and systems applications with the advantages of object orientation and the expressiveness of fuzzy sets.

The rest of this paper is organized as follows: Section 2 briefly presents the Fuzzy Logic based extensions that are needed in the object oriented model in order to represent real world situations pervaded of fuzziness. An extension of Object-Z specification language using fuzzy logic, called f-Object-L is proposed in section 3. In order to present the syntax and semantics of this extension, we show the specification of a database for a study case in section 4. Finally, section 5 remarks the conclusions of this work.

2. FUZZINESS IN OBJECT-ORIENTED CONCEPTS

Several efforts have been made in order to incorporate fuzziness to object oriented databases [2][6][9][11]. Very complete and updated description of the state of the art in fuzzy object oriented databases may be found in the books [4] and [10] that include contributions of many researchers in different issues of the topic. Nevertheless, at present time, there is neither a standard nor a formal specification language for fuzzy object oriented modeling. In the following, we analyze the object data model pointing to a way of applying fuzzy logic where needed. This analysis is made taking in mind previous fuzzy logic based extensions to object model.

Previous works sustain the idea that some object attributes have a fuzzy nature. Therefore they may be understood by means of fuzzy sets theory [3][11]. Fuzzy values are represented with possibility distributions. These distributions give for each element in base domain; the measure of how possible is this element equal to the fuzzy value. A possibility distribution is defined by $\bullet: U \rightarrow [0,1]$, where U is the base domain. For a fuzzy value, its possibility distribution corresponds to the membership function of the fuzzy set containing all its possible crisp values. This fuzzy extension for attribute level has been considered in previous related works.

Introducing fuzzy values for object attributes leads to the definition of fuzzy types. Operations for these types are defined by means of the extension principle introduced by Zadeh in [13]. This principle defines the fuzzy operator as a combination of the crisp one and the possibility distributions its operands. Results of these extended operators are fuzzy values. The existence of fuzzy values and the extension principle lead to use of a fuzzy valuated logic that is the Possibility Logic. In general case the truth value of a comparison between two fuzzy values will be a fuzzy Boolean, that is a possibility distribution on the domain {true, false}. Therefore it is necessary applying possibilistic reasoning [5].

In some real world situations, an object may partially belong to a class or several classes. For example, in a geographic information system we may define regions of risk for disaster, given an actual region, it may partially belong to this class. We propose such objects to be modeled as members of some kind of fuzzy classes. These classes must allow gradual membership of objects. This extension at object level has been considered in some previous works, other authors prefer rigid membership to classes with gradual association to super classes.

In classic object model, inheritance concept is used to express type - subtype relation that is the inclusion of a class into another. The subclass extends the concept defined by the super class adding attributes and/or methods and/or changing the object behavior for some messages. In fuzzy sets the inclusion is a gradual concept. In a fuzzy object

oriented approach, becomes necessary to be able to specify partial inclusion of a subclass to a super class. We propose here gradual inclusion between classes where each object has its own membership degree, which is different to previous approaches. In this context, we propose establishing inheritance schemes based in membership degrees.

A super class – sub class relation may be defined to be fuzzy indicating a parameter intender for the membership degree of each sub class object to the super class. At creation time it must be specified the actual membership degree for the new object. The membership degree is used to a kind of selective gradual inheritance specified in the schema with two views: Sub Class View: Defines how class inherits from its super classes and Super Class View: Indicates how sub classes inherit from this class. These views are defined by means of constraints defined in basis of the membership degree.

3. F-OBJECT-Z LANGUAGE

Object-Z is an object-oriented extension of the formal specification language Z. It adds, to Z, notions of classes and objects, and inheritance and polymorphism. By extending Z' s semantic basis, it enables the specification of systems as collections of independent objects in which self and mutual referencing are possible.

Despite the importance of formal specification languages and the richness of fuzzy logic, there are few works in formal specification of fuzzy system. Major contribution is due to Matthews and Swatman [12] who have extended the Z specification language with fuzzy logic and provided some illustrative examples. Nevertheless they have no taking in account the specification of fuzzy systems where object orientation is needed. We extend the Object-Z language with previous consideration of fuzziness in object model, proposing thus f-Object-Z.

In order to indicate the fuzzy nature of a value, we introduce the *fuzzy* type operator that defines a fuzzy data type. For example, the attribute definition Age: *fuzzy* **N** indicates that the age is a fuzzy natural number. Fuzzy terms may be defined using this operator into a constant definition scheme. Fuzzy types may be used in any place where a regular type is allowed including attributes and methods.

Due to presence of fuzziness in logic expressions it is necessary to apply possibilistic reasoning [5]. Therefore any system specified in f-Object-Z must define a constant for the inconsistency level of reasoning system. This level is the lowest satisfaction degree of a fuzzy condition intended to be consistent in the system. Obviously, this constant is a real number in [0,1]. Thus f-Object-Z extends Object-Z semantics with possibilistic reasoning.

We define new inheritance schemes in f-Object-Z that are part of a fuzzy class specification. They are the Super Class View (**Table 1**) and Sub Class View (**Table 2**). These schemes allow specification of inheritance rules based in object's membership degree to the fuzzy class. Inheritance list are identified into the scheme by the symbol •. In Super Class View we may specify constrained inheritance lists: an object is allowed to inherit things specified in the list only if its membership degree satisfies the constrain predicate. In Sub Class View we may specify a selective inheritance: an object selects things in the inheritance list whenever it satisfies the guardian predicate.

Table 1. Super Class View Scheme in f-Object-Z: At top level a variable representing the object's membership degree from instantiation in subclasses.

$\longleftarrow$ *(InheritanceDegreeVariable)*

$Predicate_1$
...
$Predicate_n$

$(Predicate_{n+1} \Leftarrow \bullet\ [Inheritance\ List_{n+1}]$
...
$(Predicate_m \Leftarrow \bullet\ [Inheritance\ List_m]$

Table 2. Sub Class View Scheme in f-Object-Z: At top level the super class name, a variable representing the membership degree of an object to the super class and others elements proper of Object-Z specification (parameters and/or rename lists).

$\longrightarrow$ *SuperClass (InheritanceDegreeVariable)…*

$Predicate_1$
...
$Predicate_k$

$(Predicate_{k+1} \Rightarrow \bullet\ [Inheritance\ List_{k+1}]$
...
$(Predicate_l \Rightarrow \bullet\ [Inheritance\ List_l]$

In f-Object-Z, an object in a fuzzy class have (possible different) membership degrees to all fuzzy its super classes. At object instantiation those degrees must be specified. We extend the Object-Z instantiation syntax with a list of membership degrees after the native class name and before regular Object-Z arguments of instantiation:

ObjectIdentifier: ClassName (MbrDegrees)… where *MbrDegrees* is the list of object membership degree to fuzzy super classes of *ClassName*. Default value for a membership degree is 1.

4. STUDY CASE

We present here the formal specification of a study case showing f-Object-Z features. Our study case is database of articles for human consumption, such as foods and medicines. In this real word situation there are various characteristics that involves user preferences an uncertainty. Things like risk of intoxication, perishable condition, tastes have a fuzzy nature. This database may be interesting for intelligent consumers that are hurried about benefits and risk of products to consume. Also producer or spender of such products may be interested in. Researchers in health and particularly in nutrition would also use a database with this information.
Possibilistic reasoning [5] impose the need of specifying the inconsistency level of reasoning system. This level must be user specified. In our study case we adopt the level 0.5. It is represented in Table 3.

Table 3. Inconsistency Level Specification in f-Object-Z

Inconsistency Level
0.5

Although presented situation is very rich in fuzziness, we will model some simple examples that illustrate the use of new features in f-Object-Z. Articles have some crisp characteristics like name, presentation and its available quantity. Articles are conformed by chemical compound that have some (possible null) toxicity level. This level may be modeled as a fuzzy attribute of the class *ChemicalCompound* (**Table 4**).

Table 4. *Chemical Compound* Class Scheme in f-Object-Z. It is a regular class.

ChemicalCompound
Toxic Level: fuzzy Real
Compound Name: seq Char

We may define some fuzzy terms for characterizing toxic level of articles or compounds. The *VeryToxic* term may be modeled in f-Object-Z as **Table 5** shows.

Table 5. *Very Toxic* Fuzzy Real Constant in f-Object-Z

VeryToxic: fuzzy R
$\forall x \mid x \le 20 \bullet \pi_{\text{Very\_Toxic}}(x) = 0$
$\forall x \mid x \ge 30 \bullet \pi_{\text{Very\_Toxic}}(x) = 1$
$\forall x \mid 20 \le x \le 30 \bullet \pi_{\text{Very\_Toxic}}(x) = 1 - \left(\frac{30 - x}{10}\right)$

We may distinguish chemical compounds that are considered to be toxins. We may model this as a subclass of *ChemicalCompound*. We state that toxins are the chemical compounds having a toxic level greater or equal to the fuzzy term *VeryToxic*. This is modeled in **Table 6**. Comparison must be performed whit the extension principle and possibilistc reasoning is applied.

Table 6. *Toxin* Class Schema in f-Object-Z.

Toxin *Chemical Compound*
$ToxicLevel \geq VeryToxic$

Another source of fuzziness is the quantity consumed of an article that is often imprecisely known. We model it with the method *ComusmeProp* of *Article* (**Table 7**) that allows a fuzzy proportion as the quantity consumed. In this context, the attribute *Quantity* of an *Article* is a fuzzy real number. It can be seen that the specification of operations involving fuzzy numbers is very easy due to extension principle.

Table 7. *Article* Class Scheme in f-Object-Z.

Article

Presentation: seq Char
Quantity: fuzzy R
Description: seq Char
Name: seq Char
Type: seq Char

Quantity ≥ 0

Consume

ConsumeQuantity?: R
ɣ(Quantity)

Quantity' = Quantity – ConsumeQuantity?
Consume_Qu antity? ≥ 0

ConsumeProp

Proportion?: fuzzy R
ɣ(Quantity)

Quantity' = Quantity (1- Proportion?)
Proportion ? ≥ 0

Regular association between articles ant theirs components is modeled in **Table 8**.

Table 8. Contains Class Scheme in f-Object-Z.

Contains

Compound: Chemical Compound
ArticleContains: Article↓
Percent: R

Articles may be considered to be toxic according to the toxic level of their components. It is modeled in the fuzzy sub class *TocxicArticle* (**Table 9**). It defines a class invariant based in a fuzzy condition that must be satisfied by all instances of this class. It also has a super class scheme with a constraint that relates the membership degree of an object with the fuzzy term *VeryToxic*.

Table 9. *Toxic Article* Class Schema in f-Object-Z.

Toxic Article

Article

Precautions: seq Char
ToxicLevel: fuzzy R

$$SUM\left\{\begin{array}{c}\forall x : Contains \mid x.ArticleContains = this \\ \bullet \left((x.Percentage) * (x.Compound) Toxic\_Level\right)\end{array}\right\} \geq VeryToxic$$

α

$$\left(\forall x : R \mid (x \geq 50 * \alpha) \bullet \pi_{VeryToxic}(x) > 0\right)$$

Some articles are perishable; we represent it by the *PerishableArticle* class of **Table 10**. Fuzzy attribute *PerishInterval* is intended for describing how an article gradually becomes damaged around *PerishDate*. *IsDamaged* method uses a given date and these attributes to compute a fuzzy truth value. In addition, in inheritance rules it is established that the lower the membership degree is, the closer to a nonperishable article must be its behavior.

Table 10. *Perishable Article* Class Scheme in f-Object-Z.

PerishableArticle

Article

PerishDate: N
PerishInterval: fuzzy R

$$\exists x : R \bullet \forall y_1 : R; y_2 : R \bullet \left(\begin{array}{l} \pi_{PerishInterval}(x) = 1 \\ \wedge \left(\begin{array}{l} (y_1 \le y_2 \le x) \Rightarrow \\ ((\pi_{PerishInterval}(x) \ge \pi_{PerishInterval}(y_2) \ge \pi_{PerishInterval}(y_2))) \end{array} \right) \\ \wedge \left(\begin{array}{l} (y_1 \ge y_2 \ge x) \Rightarrow \\ ((\pi_{PerishInterval}(x) \ge \pi_{PerishInterval}(y_2) \ge \pi_{PerishInterval}(y_1))) \end{array} \right) \end{array} \right)$$

IsDamaged

f?: N
Out!: fuzzy Bool

Out! = ((f? – PerishDate) > PerishInterval)

α

$$\exists x : R \bullet \forall y_1 : R; y_2 : R \bullet \left(\begin{array}{l} (\pi_{PerishInterval}(x) = 1) \wedge \left(x \ge \left((1-\alpha)/\alpha\right)\right) \\ \wedge \left(\begin{array}{l} (y_1 \le y_2 \le x) \Rightarrow \\ ((\pi_{PerishInterval}(x) \ge \pi_{PerishInterval}(y_2) \ge \pi_{PerishInterval}(y_2))) \end{array} \right) \\ \wedge \left(\begin{array}{l} (y_1 \ge y_2 \ge x) \Rightarrow \\ ((\pi_{PerishInterval}(x) \ge \pi_{PerishInterval}(y_2) \ge \pi_{PerishInterval}(y_1))) \end{array} \right) \end{array} \right)$$

A medicine is always a perishable and toxic article. We represent this in

Table 11.

Table 11. Medicine Class Scheme in f-Object-Z.

Medicine
Article
PerishableArticle
ToxicArticle

Medicine Type: seq Char
Requires Prescription: Boolean

On the other hand, some foods are perishable and others are not. We model food in **Table 12** as a fuzzy sub class derived from *Article* and *PerishableArticle*. It has their necessary inheritance rules into sub class view schemes. We assume a fuzzy term *high* to be defined.

Table 12. Food Class Scheme in f-Object-Z.

Food (α_1, α_2)
Article
ToxicArticle α_1

FoodType: seq Char

Food_Type $\in$ {'Meat', 'Fish', 'Cookies', 'Desserts', 'Flour', 'Chicken', 'Bread'}

Perishable Article α_2

$(\alpha_2 = high) \Rightarrow$ *(IsDamaged, PerishDate, PerishInterval)*

In previous schemes specifying our study case in f-Object-Z, it may be seen the expressive power of f-Object-Z. Gradual characteristics present in this study case may not be represented in regular object model. If one chooses to use Object-Z without the extensions the modeling of this case would be larger and more complex than presented. Moreover, some characteristics must be omitted due to rigidity. Class invariant conditions and inheritance constraints represented with f-Object-Z could not be modeled with other known fuzzy extension to the object model.

5. CONCLUDING REMARKS

We have present in this paper a new modeling tool intended for formal specification of object oriented databases in situations pervaded of fuzziness. This tool may be used to capture more semantics in some real world situations. Object Oriented Model has been analyzed proposing fuzzy set based extensions. For attribute level, we propose to allow fuzzy values represented by possibility distribution. Also fuzzy constants must be allowed. For object level, it is necessary to allow specify object gradual membership to classes. For class level, it is needed to define fuzzy subclasses and establish fuzzy inheritance rules. The Object-Z formal specification language was extended to support desired fuzzy object oriented features, generating the f-Object-Z language. Our proposal is not the only flexible extension to the object model, but we argue that this presents benefits regard other ones due to its scheme of selective flexible inheritance between classes. Our specification tool allows representing situations pervaded of uncertainty or vagueness both at object attribute level and object class level. Many of them may be more difficult to express in previous models. Furthermore, they are impossible to be modeled with classical object model. Further works will address to the revision of systems validation of correctness based in f-Object-Z specifications. It is also interesting to work in the implementation of a CASE tool based in f-Object-Z on top of an existing ORDBMS. We would like also to implement real applications using this specification tool.

Acknowledgements

I, Leonid, would like to acknowledge to Jesus Christ my Lord who have give me whole that I have and whole that I need: The intelligence and the sapience are gift form his love, but the greatest gift is the salvation that He bring us. I could never do the things that I do without the aid that He gives me every day.
We give thanks to Dr. Wíllmer Pereira who had the initial idea of this research about extension of Object Oriented Formal Specification Systems with Fuzzy Logic. Also thanks must be given to Dr. Jesús Ravelo who have introduced us to the knowledge of Z Specification Language and has walked with us in the first steps of this work.

References

[1] Bertino E., Guerrini G. Suceeding with Object Databases. John Wiley & Sons. 2001.

[2] Bordogna G., Lucarella D., Pasi, G. A Fuzzy Object Oriented Data Model. Proceedings of the IEEE 3rd International Conference of Fuzzy Systems. Vol 1, pp. 313-318. 1994.

[3] Cao, T., Creasy, P. Fuzzy types: a framework for handling uncertainty about types of objects. International Journal of Approximate Reasoning, Vol. 25, 217-253, 2000.

[4] De Caluwe R. (ed.), Fuzzy and Uncertain Object-Oriented Databases: Concepts and Models, World Scientific, 21-61, 1997.

[5] Dubois D., Prade H. Possibility Theory, Belief Revision and Non-Monotonic Logic. Technical Report. Intitut de Reserche en Informatique de Toulouse (IRIT). Université Paul Sabatier. 1993

[6] Dubois D., Prade H., Rossazza J. Vagueness, typicality and uncertainty in class hierarchies. International Journal of Intelligent Systems. Vol 6. pp. 167-183. 1991.

[7] Duke R., Rose G. Formal object-oriented specification using Object-Z. Cornerstones of computing. 2000.

[8] Duke R., Rose G., Smith G. Object-Z: A specification language advocated for the description of standards. University of Queensland. Australia. 1994

[9] Ma, Z. (ed), Advances in Fuzzy Object-Oriented Databases: Modeling and Applications, Idea Group Publishing, 2004.

[10] Ma, Z. M., Zhang, W. J., Ma, W. Y., Extending object-oriented databases for fuzzy information modeling, Information Systems, Vol. 29, Issue 5, Databases: Creation, management and utilization, 421 – 435, 2004.

[11] Marín N., Vila M.., Pons O. Fuzzy types: A new concept of type for managing vague structures. International Journal of Intelligent Systems, Vol. 15, 1061-1085, 2000.

[12] Matthews C., Swatman, P. Fuzzy Concepts and Formal Methods. Software Engineering with Computational Intelligence, Jonathan Lee (ed.), Studies in Fuzziness and Soft Computing, Vol. 121, Springer, Berlin, 2003.

[13] Zadeh L.A. The Concept of a Linguistic Variable and Its Application to Approximate Reasoning. Information Sci., 8, 199-248, 301-357; 9, 43,80. 1975.

Generación de mapas de distribución potencial de especies utilizando minería de datos y sistemas de información geográfica

Manuel F. Vargas Del Valle
Instituto Nacional de Biodiversidad (INBio), Unidad de Desarrollos Bioinformáticos
Santo Domingo de Heredia, Costa Rica, Apdo. Postal 22-3100
mvargas@inbio.ac.cr

Gabriela Marín Raventós
Universidad de Costa Rica, Escuela de Ciencias de la Computación e Informática
San José, Costa Rica, Apdo. Postal 1626-2040
gmarin@ecci.ucr.ac.cr

Resumen

La construcción de modelos de distribución de especies es uno de los problemas que se enmarcan dentro de la emergente área de investigación conocida como bioinformática. En este trabajo se propone un enfoque basado en minería de datos espaciales y sistemas de información geográfica para la generación automatizada de mapas de distribución potencial de especies. El enfoque hace uso de algoritmos de asociación mejorados con técnicas de clasificación que simplifican y optimizan el proceso de inferencia, de acuerdo con las características propias del problema. Se muestra, además, cómo la propiedad distintiva de la información espacial conocida como autocorrelación espacial puede mejorar la calidad de las predicciones. Los resultados del sistema que implementa este enfoque se muestran y se cuantifican al final.

Palabras claves: bioinformática, minería de datos, reglas de asociación, sistemas de información geográfica, distribución potencial de especies.

Abstract

Species distribution modeling is a problem considered in the emerging research field named biodiversity informatics. In this paper, an approach for the automatic generation of potential distribution maps, based on spatial data mining and geographic information systems is proposed. The approach uses association algorithms improved with classification techniques that simplify and optimize the inference process according to the distinctive characteristics of the problem. Besides, it is demonstrated how the special property of spatial information named spatial autocorrelation can improve the accuracy of the predictions. The results of the information system that implements this approach are presented and discussed at the end.

Keywords: biodiversity informatics, spatial data mining, association rules, geographic information systems, potential distribution of species.

1. INTRODUCCIÓN

El avance de la computación y su empleo en problemas propios de campos como la biología, la ecología y la geografía, han dado paso a la emergente área de conocimiento conocida como bioinformática[1], la cual aplica tecnologías de información al manejo, análisis e interpretación de datos biológicos [17].

La bioinformática está estrechamente ligada al estudio de la biodiversidad[2], que se ocupa principalmente de recolectar y analizar información para la administración científica de los recursos naturales para la conservación. Sin embargo, existe preocupación entre los conservacionistas debido a que las prácticas *ad-hoc* actuales, tales como la protección de

[1] Es importante mencionar que en el idioma inglés se diferencian los términos *bioinformatics* y *biodiversity informatics*. El primero se refiere principalmente a aplicaciones relacionadas con el nivel de genes, mientras que el otro trata con problemas en los niveles de especies y especimenes. En español, la palabra bioinformática se usa para abarcar ambas acepciones y en este trabajo nos ocuparemos de temas pertenecientes al ámbito de la segunda.
[2] Biodiversidad se define como la diversidad de genes, especies y ecosistemas que hay en la Tierra [12].

especies en peligro de extinción o el control de especies invasoras, no son adecuadas o sostenibles. Existe además un creciente nivel de concientización en el sentido de que un enfoque científico y multidisciplinario hacia la conservación ayuda a identificar mejor los posibles cursos de acción, construir confianza y proveer nuevas oportunidades [18].

Un enfoque más racional hacia el manejo de los recursos biológicos requiere de tecnologías avanzadas para el procesamiento de información. Hasta la fecha, los científicos han realizado numerosas simplificaciones y aproximaciones con el objetivo de lograr que el estudio de la biodiversidad sea posible en las plataformas computaciones existentes [18]. Una de estas aproximaciones es la generación automatizada de mapas de distribución de especies, los cuales son una herramienta de gran importancia para la visualización de información relacionada con biodiversidad. Tales mapas presentan los sitios en los cuales se han visto o se han recolectado organismos de una o varias especies. Por su parte, los mapas de distribución potencial permiten estimar, mediante algún tipo de inferencia, cual sería la distribución de una o varias especies en un área en la cual aún no se cuenta con datos suficientes para presentar un mapa de distribución basado en observaciones o recolecciones. Los mapas de distribución potencial presentan gráficamente las áreas en donde existe una probabilidad aceptable de que la especie esté presente. En la figura 1 se muestra un mapa de distribución y un mapa de distribución potencial.

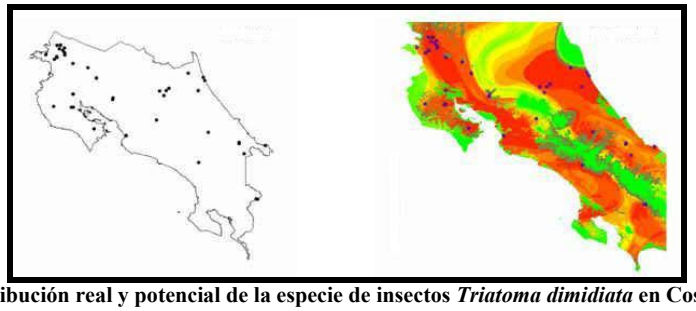


Figura 1. Mapas de distribución real y potencial de la especie de insectos *Triatoma dimidiata* en Costa Rica, transmisora del mal de chagas. Las áreas más rojas (oscuras) del mapa de distribución potencial (el de la derecha) son las que tienen una mayor probabilidad de presencia de la especie.

Usualmente, los mapas de distribución potencial se confeccionan con base en factores conocidos *a priori* que describen el comportamiento de las especies y por ende su distribución potencial. Algunos de los factores que pueden ser tomados en cuenta para estimar la distribución de una especie son, entre otros, la historia geológica, el clima, el microclima, la disponibilidad del alimento, la química del ambiente y la competencia [3]. Otras variables ambientales muy comúnmente usadas para estos efectos son altitud, temperatura, precipitación y tipo de suelo, además de combinaciones o variaciones entre las mismas (e.g. precipitación media del mes más caliente o temperatura máxima del mes más húmedo). Para algunos casos, se considera común el uso de 30 o más variables diferentes [19]. Dada la gran cantidad de elementos que pueden tener que considerarse para confeccionar un mapa de distribución potencial, la elaboración de éstos puede ser sumamente compleja.

Una propiedad importante de los mapas de distribución potencial es que permiten realizar inferencias no solo acerca de la especie referida en el mapa, sino también de otras con características similares o que estén relacionadas entre sí de alguna forma (e.g. predadores-presas o huéspedes-parásitos). En zonas de alta biodiversidad, tales como los territorios de muchos países latinoamericanos, esta característica puede ser empleada para predecir el hábitat de grupos no muy bien estudiados a partir de la información de otros mejor conocidos, ya que la sola tarea de listar todas las especies presentes en un área es excesivamente extensa, costosa y puede producir resultados poco precisos.

Se han desarrollado varios enfoques estadísticos y computacionales que permiten modelar la distribución de una especie con base en técnicas como regresión logística, redes neuronales artificiales, árboles de decisión y algoritmos genéticos [15,16]. En la mayoría de ellos, se pretende extrapolar los sitios en donde se tiene certeza de que habita la especie hacia zonas en donde su distribución es desconocida. Esta extrapolación se realiza con base en variables ecológicas y ambientales como las mencionadas anteriormente.

En años recientes, el área de investigación denominada como minería de datos ha surgido como una alternativa a enfoques tradicionales para el análisis de datos y el descubrimiento de información. En particular, el mejoramiento y la popularización de la minería de datos y su integración con los sistemas de información geográfica o GIS (*Geographic Information System*) han propiciado el desarrollo de la minería de datos espaciales, la cual combina ambas áreas de investigación y proporciona la posibilidad de extraer conocimiento que no está almacenado explícitamente en bases de datos que contienen información espacial. En la actualidad, es posible tener acceso gratuito, por medio de Internet a

bases de datos que contienen millones de registros de información espacial relacionada con biodiversidad (e.g. Global Biodiversity Information Facility (GBIF), http://www.gbif.net; Atta, http://www.atta.inbio.ac.cr; TROPICOS, http://mobot.mobot.org/W3T/Search/vast.html; Mammal Networked Information System (MaNIS), http://elib.cs.berkeley.edu/manis/); las cuales pueden ser utilizadas para la confección de modelos de distribución de especies.

El enfoque tradicional para extraer conocimiento de las bases de datos espaciales ha sido a través de la estadística espacial [6]. Sin embargo, ésta presenta el inconveniente de asumir, al igual que la estadística clásica, que las muestras de datos son generadas al azar o aleatoriamente, lo cual no es aplicable a la información espacial, en donde los datos están altamente autocorrelacionados [4]. En los años recientes, se han desarrollado varias técnicas de minería de datos espaciales que han sido derivadas a partir de la investigación realizada en minería de datos convencional y que toman en consideración las características particulares de la información espacial [7,8, 9,21].

En este documento se presenta un enfoque para la generación de mapas de distribución potencial de especies basado en reglas de asociación espaciales, una técnica derivada de su conocida contraparte de la minería de datos clásica. Asimismo, se pretende mostrar que la minería de datos clásica, que usa reglas de asociación, y la teoría de modelaje de especies pueden ser enriquecidas con el concepto de autocorrelación espacial para así lograr predicciones más precisas.

Primeramente, se desarrollarán brevemente los conceptos más importantes del marco teórico, posteriormente se explicará la metodología del enfoque y por último se mostrará y analizará un ejemplo de su aplicación.

2. MARCO TEÓRICO

Como se mencionó, el enfoque que se ha desarrollado se enmarca dentro del área de investigación denominada como minería de datos. La minería de datos, también llamada descubrimiento de conocimiento en bases de datos, es un proceso no trivial de extracción de información previamente desconocida y potencialmente útil (tal como reglas de conocimiento, limitantes, patrones, etc.) realizado a partir de los datos contenidos en bases de datos [13]. Mediante este proceso, es posible extraer de conjuntos de datos relevantes, información de alto nivel que puede ser investigada desde diferentes ángulos. Esto permite que las grandes bases de datos funcionen como fuentes confiables de generación y verificación del conocimiento [5].

En la actualidad, mucha información contenida en bases de datos es información espacial, la cual ha sido capturada y procesada con sistemas de información geográfica. Un Sistema de Información Geográfica o GIS es un sistema de información diseñado para trabajar con datos referenciados según su localización geográfica. Un GIS describe el mundo real en términos de [14]:

- Localización: posiciones geográficas.
- Atributos: características que no están directamente relacionados con la posición, tales como tipo de suelo, altitud y datos demográficos.
- Relaciones espaciales: como están relacionados los objetos entre sí.
- Información temporal: la evolución en el tiempo de los datos espaciales y no espaciales.

El mejoramiento y la popularización de las áreas relacionadas con la minería de datos y los sistemas de información geográfica han propiciado el desarrollo de técnicas que permiten la combinación de ambas áreas de investigación. Nace así la minería de datos espaciales, o descubrimiento de conocimiento en bases de datos espaciales, la cual se refiere a la extracción de conocimiento implícito, relaciones espaciales u otros patrones que no están almacenados explícitamente en los datos espaciales [1].

Como ya se mencionó, las técnicas de estadística y minería de datos espaciales tienden a asumir que las muestras son obtenidas aleatoriamente, lo cual es un supuesto que viola la primera ley de la geografía:

> "*Todo está relacionado con todo, pero las cosas cercanas están más relacionadas que las cosas distantes*" [20].

En otras palabras, los atributos de los objetos espaciales cercanos tienden a afectarse sistemáticamente unos a otros. En estadística espacial, un área de la estadística dedicada al análisis de los datos espaciales, esto se denomina *autocorrelación* [6]. Las técnicas de descubrimiento de conocimiento que ignoran la autocorrelación espacial suelen tener un rendimiento deficiente en la presencia de datos espaciales [4]. Por lo anterior, es importante que las técnicas de minería de datos aplicadas a datos espaciales tomen en consideración las características particulares de éstos.

En el enfoque presentado en este trabajo se explicará como el concepto de autocorrelación espacial puede ser introducido en la generación de reglas de asociación.

3. DEFINICIÓN DEL PROBLEMA Y ENFOQUE PROPUESTO

El problema de predecir los sitios en donde podría encontrarse una determinada especie es un caso particular del problema de predicción de localizaciones, el cual puede describirse informalmente como el intento de predecir la ubicación en donde tiene o tendrá lugar un fenómeno determinado. Esta predicción se realiza con base en un conjunto de variables que afectan el fenómeno y en observaciones previas del mismo.

3.1 Definición formal del problema

Seguidamente, se presenta la definición formal del problema de predicción de localizaciones, la cual está basada en el trabajo de Chawla y otros [4].

Dados:

1. Un marco de referencia espacial S con sitios $\{s_1, ..., s_n\}$ ubicados en un espacio geográfico.
2. Una colección de funciones explicatorias $f_{X_k} : S \rightarrow R^k$, $k = 1...K$, en donde R^k es el rango de posibles valores de la k-ésima función.
3. Una función $f_Y : S \rightarrow R^Y$, en donde $R^Y = \{0, 1\}$.

El objetivo es encontrar:

a. una familia F de funciones de predicción: $R^1 \times ... \times R^K \rightarrow R^Y$,
b. y una función $f'^Y \varepsilon F$ tal que se maximice la función:
similaridad($mapa(f'^Y(f_{X_1}, ..., f_{X_k}))$, $mapa(f_Y)$).

A continuación se detalla como los elementos de este planteamiento corresponden con los del problema de generación de mapas de distribución potencial de especies:

- El marco de referencia espacial S corresponde a un conjunto de celdas o polígonos en los que se divide el espacio geográfico en estudio para determinar si en cada uno de los mismos se encuentra presente la especie. Estas celdas se generan al sobreponer una cuadrícula sobre el mapa respectivo. Los mapas divididos mediante cuadrículas se conocen como mapas en formato ráster.
- Las funciones explicatorias f_{X_k} corresponden a variables ambientales o niveles temáticos del espacio geográfico en el que se está realizando el estudio: altitud, precipitación, ecosistemas, tipos de suelo, etc. Es importante notar que el rango R^k de posibles valores es diferente para cada función y en algunos casos estos valores pueden ser números reales y en otras categorías. Esto es una diferencia con respecto al modelo propuesto por Chawla y otros, que asumen que el dominio de estas funciones es siempre el conjunto de los números reales.
- La función f_Y indica si la especie se encuentra presente en un sitio determinado. El dominio R^Y de esta función es igual a {0, 1}, en donde 0 denota ausencia y 1 denota presencia. Los valores de esta función provienen de recolecciones u observaciones de la especie.
- Cada uno de los miembros de la familia F de funciones de predicción combina valores de las funciones explicatorias (coberturas geográficas) que determinan si la especie se presenta dada esa combinación.
- La función f'^Y es el subconjunto de F que genera un mapa lo más similar posible al correspondiente a f_Y.

Como puede observarse, en este caso, la predicción se realiza al tratar de maximizar la similitud entre dos mapas: el mapa con las ocurrencias predichas y el mapa que presenta las observaciones o recolecciones de la especie. La similitud se mide al contar los sitios cuyo valor de presencia o ausencia coincide en ambos mapas.

3.2 El enfoque propuesto

Para la resolución del problema de predicción de localizaciones en el contexto de la generación de mapas de distribución potencial de especies, se ha diseñado un enfoque basado en la minería de reglas de asociación.

El problema de la minería de reglas de asociación se define formalmente con base en los siguientes conceptos [2]:

- Sea $I = \{i_1, i_2, \ldots, i_m\}$ un conjunto de literales llamados ítems.
- Sea D un conjunto de transacciones en donde cada transacción T es un conjunto de ítems de manera tal que $T \subseteq I$.

Además,

- Una regla de asociación es una implicación de la forma $X \Rightarrow Y$, en donde $X \subseteq I$, $Y \subseteq I$ y $X \cap Y = \varnothing$.
- La regla $X \Rightarrow Y$ tiene confianza c en el conjunto de transacciones D si c% de las transacciones en D que contienen a X contienen también a Y.
- La regla $X \Rightarrow Y$ tiene soporte s en D si s% de las transacciones en D contienen $X \cup Y$.

Dado un conjunto D de transacciones, el problema de la minería de reglas de asociación consiste en generar todas las reglas de asociación que tengan valores de confianza y soporte mayores que los de confianza mínima (*minconf*) y soporte mínimo (*minsup*), los cuales son especificados por el usuario. Estas definiciones son neutrales con respecto a la representación de D, que para efectos de implementación puede ser una tabla relacional, un archivo de datos o el resultado de una expresión relacional [2].

Originalmente, la minería de reglas de asociación está inspirada en el análisis de datos de canastas de supermercado (*basket data*) y, en general, de operaciones financieras y comerciales. Un ejemplo de regla de asociación puede ser que el 70% de las personas que se hospedan un hotel utilizan también su servicio de restaurante. Sin embargo, la facilidad con la que las reglas de asociación pueden ser comprendidas y expresadas en lenguaje natural, hace que sea posible aplicarlas en una gran variedad de contextos.

Uno de estos contextos es el de la información espacial, que da paso a la minería de reglas de asociación espaciales. Una regla de asociación espacial es aquella que describe la implicación de una característica o de un conjunto de características en otro conjunto de características dentro del contexto de una base de datos espacial. Por ejemplo, una regla como "la mayoría de las ciudades grandes de Canadá están cerca de la frontera entre Estados Unidos y Canadá" es una regla de asociación espacial [10].

El enfoque de este trabajo consiste en generar reglas de asociación espaciales que puedan ser utilizadas en la confección de mapas de distribución potencial de especies, un caso particular del problema de predicción de localidades. Las reglas de asociación reflejan la relación que hay entre las variables ambientales y la especie cuya distribución se quiere predecir.

Así, con base en la definición del problema de predicción de localidades anteriormente dada, se desarrolló un método que genera:

a) Una familia F de funciones de predicción: $R^1 \times \ldots \times R^K \rightarrow R^Y$, las cuales están expresadas como reglas de asociación espaciales que tienen la forma:

$$f_{X1}(s) = y_1 \wedge f_{X2}(s) = y_2 \wedge \ldots \wedge f_{Xk}(s) = y_k \rightarrow f_Y(s) = y_Y \quad (t, c),$$

que equivaldría a, por ejemplo:

$$altitud()=1000\text{-}2000 \wedge uso\_de\_la\_tierra()=pastizal \wedge \ldots \wedge temperatura()=alta \rightarrow presencia()=1 \quad (0.4, 0.6),$$

en donde t y c corresponden a los valores de soporte y confianza explicados anteriormente.

b) Una función $f'^Y \varepsilon F$ tal que se maximice la función:
similaridad($mapa(f'^Y(f_{X1}, \ldots, f_{Xk}))$, $mapa(f_Y)$).

Esta función f'^Y es un subconjunto de reglas de asociación extraído del conjunto de reglas mencionado en a) y al que de aquí en adelante se le llamará ***clasificador***. El clasificador contendrá las reglas más apropiadas para maximizar la similaridad entre el mapa de predicción y el mapa de presencias/ausencias generado a partir de recolecciones u observaciones.

3.3 Desarrollo del enfoque propuesto

El método desarrollado toma como entrada:

1. Un conjunto de mapas en formato ráster correspondientes a variables ambientales. Cada celda del mapa contiene un valor de la variable (e.g. precipitación media anual, altitud).
2. Un mapa en formato ráster que indica la presencia o ausencia de la especie en cada celda.
3. Un valor de soporte mínimo para las reglas generadas.

4. Un valor de confianza mínima para las reglas generadas.

La salida es otro mapa en formato ráster que muestra la distribución potencial de la especie.

Los valores contenidos en los mapas deben transformarse a un formato adecuado para ser procesados mediante algoritmos de minería de reglas de asociación y así obtener un clasificador. El clasificador resultante es transformado en otro mapa en formato ráster que contiene en cada celda la probabilidad de que la especie esté presente en ella.

Gráficamente, la metodología general de predicción puede expresarse como en la figura 2.

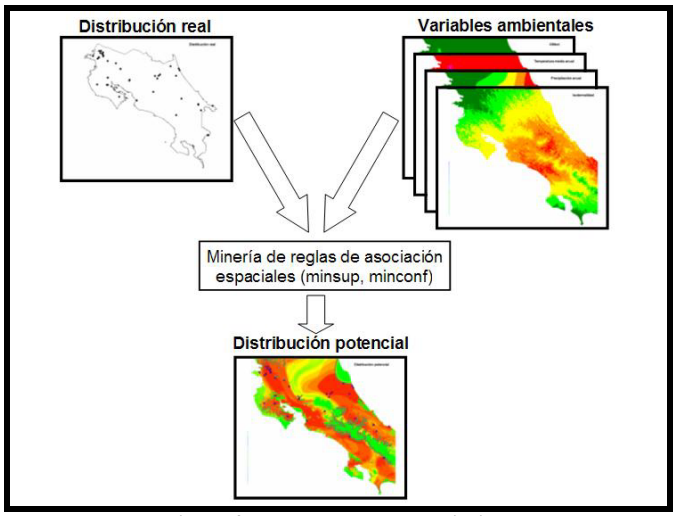


Figura 2. Metodología de predicción.

Más detalladamente, la metodología se divide en cinco grandes etapas:

1. Preparación de los mapas de entrada.
2. Transformación de los mapas de entrada en un formato adecuado para ser procesado por los algoritmos de reglas de asociación.
3. Aplicación de los algoritmos de reglas de asociación para obtener un clasificador.
4. Generación del mapa de predicción a partir del clasificador.
5. Validación del modelo de predicción obtenido.

Seguidamente, se describe en detalle cada una de las etapas.

3.3.1 Preparación de los mapas de entrada

Para preparar los mapas se siguen los siguientes pasos:

1. Las coordenadas geográficas de los sitios de recolección y de los niveles temáticos de los datos ambientales se proyectan a un mismo sistema de coordenadas (e.g. WGS84 Latitud/Longitud, UTM).
2. Los niveles temáticos de los datos ambientales se convierten a un formato ráster con una misma resolución. Por ejemplo, celdas de 1 minuto cuadrado.
3. El mapa de presencia/ausencia se convierte también a un formato ráster en el que cada celda tiene un valor de 0 para ausencia y 1 para presencia. La resolución debe ser igual a la de los niveles temáticos de las variables ambientales.
4. Si los niveles temáticos corresponden a datos continuos, deben "discretizarse" para poder ser procesados por los algoritmos de generación de reglas de asociación.
5. Se crea un nivel temático para aplicar el concepto de autocorrelación espacial, según el cual las probabilidades de ocurrencia de una especie en un área se incrementan si hay ocurrencias de la misma especie en áreas cercanas. Se utiliza una medida de contagio para cada celda, con base en un entorno de tres órdenes, para estimar una probabilidad de ocurrencia basada en distancias. El ***contagio*** es calculado

como un promedio ponderado del número de celdas ocupadas entre un conjunto de k_a vecinas de una celda central y_a, de manera que según la definición de Segurado y Araujo [16]:

$$\text{Contagio} = \sum_{b=1}^{ka} w_{ab} y_b / \sum_{b=1}^{ka} w_{ab}$$

en donde el peso asignado a la celda y_b es $w_{ab} = 1/d_{ab}$ y d_{ab} es la distancia entre las celdas y_a y y_b. Se utiliza un entorno de tres órdenes (d = 1,2,3). Las vecinas del primer orden son las ocho celdas adyacentes a la celda central, las del segundo orden las dieciséis que rodean a las de primer orden y las del tercer orden las treinta y dos que rodean a las del segundo orden.

En la figura 3 se presenta un ejemplo de un mapa de ocurrencias para las cuales se ha aplicado la fórmula de contagio:

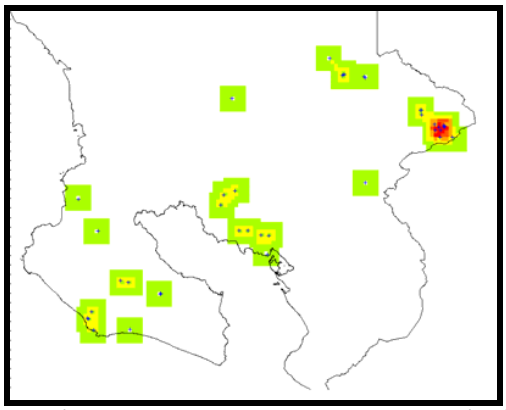


Figura 3. Aplicación de la formula de contagio. Puede observarse que las celdas más rojas (oscuras) son las que tienen más presencias en sus alrededores.

La fórmula se aplica mediante las facilidades algebraicas para datos ráster proporcionadas por el sistema de información geográfica GRASS. El mapa resultante permite apreciar el grado de influencia que recibe cada celda de las celdas próximas en las cuales se han registrado ocurrencias de la especie. Una vez que esta capa se ha discretizado en tres niveles de contagio: alto, medio y bajo, se usa como otro nivel temático de entrada.

3.3.2 Transformación de los mapas de entrada en un formato adecuado para ser procesado por los algoritmos de reglas de asociación

Los valores expresados en los mapas deben ser transformados en los elementos que manejan los algoritmos de reglas de asociación. Estos son ítems y transacciones, tal y como fueron definidos arriba. Con este fin, la información de las celdas del k-ésimo nivel temático (k = 1…K) se coloca en una tabla cuyas tuplas tienen la forma:

(#celda, valor_nivel_temático_k, etiqueta)

en donde el valor es un número entero que corresponde a la categoría resultante del proceso de discretización anteriormente mencionado, y la etiqueta es un campo textual que describe el valor correspondiente. Cabe aclarar aquí que estas tablas no se normalizan, debido a que los programas de manejo de información geográfica que se emplearon las utilizan en el formato descrito, por motivos de rendimiento y despliegue gráfico.

De manera similar, la información correspondiente a los sitios de recolección de cada especie se coloca también en tablas cuyas tuplas tienen la forma:

(#celda, valor_presencia_ausencia, etiqueta)

En este caso, el valor puede ser solamente 0 (ausencia) ó 1 (presencia).

La tabla de contagio tiene la misma estructura. Su valor es un número real del intervalo [0…3] y se discretiza de la siguiente manera:

0 – 0.1 = 1 (Contagio bajo)

0.1 – 0.4	=	2 (Contagio medio)
> 0.4	=	3 (Contagio alto)

Posteriormente, se reúne la información para cada celda y se forma una tabla cuyas tuplas tienen la forma:

(#celda, valor_nivel_tematico_1, ..., valor_nivel_tematico_K, valor_contagio, valor_presencia_ausencia)

Según la terminología de los algoritmos de generación de reglas de asociación, cada valor de cada nivel temático reflejado en la tabla anterior corresponde a un *ítem* y cada combinación de valores (o sea, cada registro) corresponde a una *transacción*.

Este conjunto de transacciones, junto con los valores minsup y minconf que especifica el usuario, sirven de entrada a los algoritmos de generación de reglas de asociación. Pero antes, es necesario dividir los datos en dos conjuntos: uno para construir el modelo de predicción y otro para evaluarlo.

3.3.3 Aplicación de los algoritmos de reglas de asociación para obtener un clasificador

Con el fin de poder evaluar objetivamente la exactitud de las predicciones, las transacciones de entrada deben dividirse en dos conjuntos:

- **Entrenamiento**: Se usa para generar las reglas.
- **Evaluación**: Se usa para aplicar las reglas y comprobar su exactitud.

Lo que se busca es que el algoritmo pueda "aprender" del conjunto de datos de entrenamiento para posteriormente aplicar ese aprendizaje en el conjunto de datos de evaluación y medir su efectividad. La efectividad se mide al contar el número de aciertos y errores que produce el clasificador. Es de vital importancia que ambos conjuntos sean independientes, para así garantizar que las mediciones hechas en el conjunto de evaluación no estén sesgadas.

Ahora, se necesita un conjunto de entrenamiento grande para obtener el mejor clasificador posible y se necesita también un conjunto de evaluación grande para medir con la mayor precisión posible la efectividad del modelo de predicción. Esto no representa un gran problema cuando se cuenta con poblaciones de datos grandes, pero este, por lo general, no es el caso con los datos de recolecciones u observaciones biológicas, especialmente en lo que se refiere a datos de presencias.

Con el fin de maximizar el tamaño de los conjuntos de entrenamiento y evaluación sin detrimento de la independencia de ambos, se emplea una técnica estadística llamada validación cruzada (*cross validation*), de acuerdo con la cual los datos se dividen en N conjuntos disjuntos. Luego, alternadamente, cada uno de estos N conjuntos funciona como conjunto de evaluación y la unión de los N-1 restantes funciona como conjunto de datos de entrenamiento, por lo que el procedimiento se repite N veces. El producto final es el promedio de los resultados arrojados por los N conjuntos de evaluación. Nótese que la elección de N es arbitraria, pero por lo general es un número comprendido entre 3 y 10.

Los N conjuntos se generan mediante un muestreo aleatorio con reemplazo y cada uno contiene un 50% de datos de ausencia y un 50% de datos de presencia, a menos que se especifique de otra manera. El muestreo con reemplazo permite que cada sitio pueda ser elegido varias veces, lo que con frecuencia es necesario para que los conjuntos tengan el tamaño suficiente para que se generen reglas útiles a partir de ellos. El esquema de validación cruzada también permite minimizar el peligro de elegir una muestra no representativa en los conjuntos de datos de entrenamiento y evaluación.

En el conjunto de datos de entrenamiento, las reglas son generadas por medio del algoritmo Apriori, que es uno de los más empleados en minería de reglas de asociación [2]. Sin embargo, con el fin de adaptar el algoritmo a las necesidades particulares de este problema, dos modificaciones importantes fueron introducidas, con base en una variante de Apriori desarrollada por Liu y otros [11].

La primera variante modifica el algoritmo para que genere solamente reglas de la forma:

$condición\ \Lambda\ condición\ \Lambda\ \ldots\ \Lambda\ condición \rightarrow presencia()=0$
o
$condición\ \Lambda\ condición\ \Lambda\ \ldots\ \Lambda\ condición \rightarrow presencia()=1$

Esta adaptación se realizó para descartar todos los consecuentes cuyos valores no pertenezcan a {0, 1}, ya que el algoritmo Apriori original genera reglas para todos los posibles consecuentes, los cuales son todas las coberturas

geográficas que se estén considerando. Esto puede verse como una integración entre minería de reglas de asociación y minería de reglas de clasificación, que es un tema comentado por varios autores [11]. La diferencia entre los dos tipos de reglas es que en asociación el objetivo del descubrimiento no está predeterminado, mientras que en clasificación sí lo está. El objetivo de este tipo de integración es construir clasificadores con un rendimiento superior al que brindan los algoritmos de clasificación tradicionales como C4.5. Para el caso en estudio, es claro que interesan solamente las reglas cuyo consecuente indique presencia o ausencia de la especie, así que sí existe un objetivo predeterminado.

El resultado de la aplicación de esta primera variante es entonces un conjunto de reglas $R = \{r_1, r_2, \ldots, r_n\}$, cuyos consecuentes indican presencia o ausencia. Produce, además, dos efectos importantes:

a. Se reduce el número de reglas. Esto es de suma importancia, ya que en este tipo de algoritmos el número de reglas crece exponencialmente con respecto al número de ítems (o valores de coberturas geográficas, en este caso).
b. Se comienzan a filtrar las reglas que son potencialmente más útiles o interesantes. Este filtrado se extiende en una etapa posterior, como se explica a continuación.

La segunda variante consiste propiamente en la construcción de un clasificador a partir de las reglas espaciales generadas y que minimice el número de errores con respecto a la predicción del objetivo, que en este caso es presencia/ausencia de la especie. Esta es una etapa que el algoritmo Apriori original no contempla por las características ya explicadas de la minería de reglas de asociación. Para construir el mejor clasificador sería necesario evaluar en el conjunto de datos de entrenamiento todos los posibles subconjuntos de reglas y seleccionar el subconjunto con la secuencia que genere el menor número de errores. Para n reglas, existen 2^n subconjuntos, lo que puede resultar en un número muy alto de combinaciones para evaluar, sin mencionar todas las posibles secuencias. Liu y otros desarrollaron un algoritmo heurístico para construir el clasificador que funciona bastante bien en comparación con algoritmos de clasificación tales como C4.5 [11], el cual fue incorporado en este enfoque.

3.3.4 Generación del mapa de predicción a partir del clasificador

El clasificador obtenido es de la forma $<r_1, r_2, \ldots, r_m>$ en donde $r_i \in R$, $m <= n$ y r_i tiene precedencia sobre r_{i+1} de acuerdo a un orden definido que se explicará más adelante. Luego, las reglas de este clasificador son aplicadas en el conjunto de datos de entrenamiento, comenzando por r_1 y clasificando de acuerdo al consecuente (ausencia/presencia) todas las transacciones (sitios) que satisfagan el antecedente (variables ambientales) de la regla. Posteriormente, r_2 es aplicado en las transacciones restantes y así se sigue hasta que se terminen las reglas o los datos del conjunto de entrenamiento.

Una vez terminado este proceso, los sitios del conjunto de datos de entrenamiento se han clasificado de acuerdo a una predicción de ausencia o presencia a la cual se le han asignado además valores de soporte y confianza. El valor de soporte puede utilizarse para conocer el grado de cobertura de la regla en la población. Un valor de soporte alto indica que la regla se aplica en muchos sitios y un valor bajo indica que se aplica en pocos sitios. Por su parte, la confianza especifica la probabilidad de que la predicción se cumpla para el sitio. La probabilidad de ocurrencia de la especie en un sitio particular está dada por la confianza misma en el caso de que se prediga presencia y por (1 - confianza) si se predice ausencia.

Este proceso de aplicación del clasificador cumple con dos características importantes:

a. La regla que es aplicada a cada sitio es la que tiene la más alta prioridad de todas las que son aplicables. La prioridad de las reglas se establece de acuerdo al siguiente orden:
    i. Presencia antes que ausencia. Si un sitio cumple las condiciones tanto para una regla de presencia como para una de ausencia, se aplica la de presencia. Este criterio fue agregado para efectos de este trabajo y no está contemplado en la definición original del algoritmo, que se basa en los dos criterios siguientes. La razón por la cual se favorecen las reglas de presencia se explica en la sección correspondiente a la validación del modelo de predicción
    ii. Mayor confianza.
    iii. Mayor soporte.

b. Se eliminan aquellas reglas que no aportan más precisión al clasificador. La precisión se mide con base en el número de aciertos y errores, o sitios bien o mal predichos, que genera cada regla.

De esta manera, se obtiene un clasificador que minimiza el número de errores y tiene el menor tamaño posible, lo que lo hace más rápido de procesar y más fácil de interpretar. Una vez que se han generado las probabilidades de ocurrencia para cada uno de los sitios, puede construirse un mapa ráster con esa información.

3.3.5 Validación del modelo de predicción obtenido

Para validar el modelo obtenido, el clasificador se aplica en el conjunto de datos de evaluación y en el total de la población. Para cada conjunto se obtienen cuatro valores:

a = Número de sitios en donde se predijo presencia y el valor real es presencia.
b = Número de sitios en donde se predijo ausencia y el valor real es ausencia.
c = Número de sitios en donde se predijo ausencia y el valor real es presencia.
d = Número de sitios en donde se predijo presencia y el valor real es ausencia.

Es fácil observar que un buen modelo de predicción maximizaría a y b, que pueden considerarse como aciertos, mientras que minimizaría c y d, que serían los errores. Entonces, la precisión del modelo en cada conjunto está dada por:

Precisión = (a + b)/(a + b + c + d)

Este valor refleja el grado de similitud entre el mapa con las ocurrencias de la especie y el mapa de predicción. Mientras más cercano a uno sea el valor de precisión, más similares son ambos mapas y mientras más cercano a cero, más diferentes.

A primera vista puede parecer que los modelos deben tratar de maximizar el valor de precisión. Sin embargo, es importante también tener en cuenta que los *errores de omisión*, los del tipo c, son más graves que los *errores de comisión*, los del tipo d. Esto se debe a que los datos de ausencia no son, por lo general, datos comprobados exhaustivamente por el trabajo de recolección u observación en el campo. El trabajo de recolección rara vez corresponde a experimentos bien diseñados, más bien es de carácter oportunístico, ya que los recolectores acostumbran realizar su trabajo en las regiones donde cuentan con facilidades de tipo logístico. Así que un dato de ausencia no necesariamente significa que la especie no esté presente en el sitio. Puede significar también que no ha sido visitado o que se visitó en una época del año en la que la especie no es visible por alguna razón como floración o migración, por ejemplo.

Por el contrario, los datos de presencia proporcionan una evidencia más contundente, ya que una recolección o un avistamiento sí pueden dar certeza de que la especie se encuentra presente en el sitio (o al menos de que se encontraba presente en el momento de la recolección). Esta es la razón por la cual las reglas de presencia tienen una prioridad más alta a la hora de construir el clasificador. Por lo tanto, a la hora de evaluar el modelo de predicción construido, es aconsejable buscar una baja proporción de errores de omisión.

4. EJEMPLO DE APLICACIÓN DEL ENFOQUE

Para ilustrar el enfoque se generó un modelo de predicción para Costa Rica de la especie de mariposas *Anticla antica*. El algoritmo de generación de reglas de asociación espaciales descrito anteriormente se implementó en un prototipo de sistema de información en el lenguaje Java, con la herramienta NetBeans (http://www.netbeans.org), y para el manejo de los mapas raster se utilizó el sistema de información geográfica GRASS (*Geographical Resource Analysis Support System,* http://grass.itc.it/). Ambos programas se conectaron a una base de datos PostgreSQL (http://www.postgresql.org/), la cual provee facilidades especiales para el manejo de datos geográficos. Todas las herramientas son de tipo *Open Source* y corren sobre sistema operativo Linux en un computador PC.

Como datos de entrada se utilizaron 128 sitios de recolección diferentes, distribuidos en todo el territorio costarricense, junto con información de altitud, temperatura media anual y precipitación anual. La información de recolecciones se tomó del sitio web del sistema *Atta* del Instituto Nacional de Biodiversidad de Costa Rica (INBio) (http://www.atta.inbio.ac.cr), y la información geográfica del sitio del proyecto WorldClim de la Universidad de Berkeley (http://biogeo.berkeley.edu/worldclim/worldclim.htm). Tanto las coberturas geográficas como los puntos de recolección se trabajaron con el sistema de coordenadas WGS84 Longitud/Latitud. Se utilizó una resolución de 30 x 30 segundos (aproximadamente 1 km^2) en las coberturas geográficas.

Según el esquema de validación cruzada, los datos de entrada se dividieron en 4 subconjuntos, cada uno de los cuales funcionó alternadamente como conjunto de evaluación. Cada conjunto tenía 500 sitios de presencia y 500 de ausencia y fueron obtenidos mediante muestreo aleatorio con reemplazo. Se generaron reglas de asociación con una confianza de

0.6 y un soporte de 0.001. Con el fin de medir el efecto de la autocorrelación espacial en el proceso, este se ejecutó tanto con como sin la capa geográfica correspondiente. Los resultados se muestran en el siguiente cuadro:

		Resultados en el conjunto de datos de evaluación						*Proyección en la totalidad de los datos*					
						Precisión						*Precisión*	
	Rotación	**a**	**b**	**c**	**d**	**Total**	**Presencias**	**a**	**b**	**c**	**d**	**Total**	**Presencias**
Sin autocorrelación espacial	1	398	257	102	233	0.66	0.80	110	39070	18	34497	0.53	0.86
	2	399	291	101	200	0.70	0.80	108	40508	20	32150	0.56	0.84
	3	415	254	85	232	0.68	0.83	115	36079	13	37925	0.49	0.90
	4	381	298	103	179	0.71	0.76	106	43386	21	27413	0.61	0.83
	Promedio	**398**	**275**	**97.8**	**211**	**0.69**	**0.80**	**110**	**39761**	**18**	**32996**	**0.55**	**0.86**
						Precisión						*Precisión*	
	Rotación	**a**	**b**	**c**	**d**	**Total**	**Presencias**	**a**	**b**	**c**	**d**	**Total**	**Presencias**
Con autocorrelación espacial	1	383	370	106	128	0.76	0.77	115	53644	12	19641	0.73	0.90
	2	463	258	13	241	0.74	0.93	121	37909	5	35733	0.52	0.95
	3	417	319	83	177	0.74	0.83	118	45493	10	27875	0.62	0.92
	4	418	286	53	198	0.74	0.84	120	41579	6	28481	0.59	0.94
	Promedio	**420**	**308**	**63.8**	**186**	**0.74**	**0.84**	**119**	**44656**	**8.25**	**27933**	**0.62**	**0.93**

Nótese que si bien es cierto la precisión total (que incluye ausencias y presencias) es relativamente baja en ambos casos, las precisiones de presencias, las cuales son más importantes por las razones expuestas anteriormente, son mucho mejores. Sin el uso de la capa geográfica de autocorrelación espacial, el algoritmo predijo correctamente el 80% de los sitios de presencia en el conjunto de datos de evaluación y el 86% en la totalidad de los datos. Mediante el uso de la autocorrelación espacial, la predicción mejoró a un 84% en el conjunto de evaluación y a un 93% en el conjunto completo. El tiempo de ejecución para la predicción que no utilizó autocorrelación espacial fue de 23 minutos y el de la que sí usó autocorrelación espacial fue de 27 minutos. Ambos situaciones son intensivas en cálculo, como es normal en estos casos, y la incorporación de la autocorrelación espacial no aumenta significativamente el tiempo de ejecución.

Para medir la significatividad estadística de las predicciones, se utilizó la prueba chi cuadrado (Chi2), la cual es una prueba no parámetrica usada frecuentemente en análisis bivariados y que permite estimar un nivel de confianza para aceptar o rechazar una hipótesis. La predicción sin autocorrelación espacial promedió un Chi2 de 18.545, correspondiente a un valor de P igual a 0.00002. Para la predicción con autocorrelación espacial los valores de Chi2 y P fueron 25.59 y 0.0000004 respectivamente. El valor de P puede interpretarse como la probabilidad de que la calidad de la predicción sea producto del azar. Puede entonces observarse que los resultados de ambas predicciones son estadísticamente significativos, pero los de la predicción que utiliza autocorrelación espacial lo son aún más.

El mapa de distribución potencial para la especie, que se obtiene al sumar con álgebra ráster los 4 modelos parciales que usaron autocorrelación espacial, es el que se muestra en la figura 4.

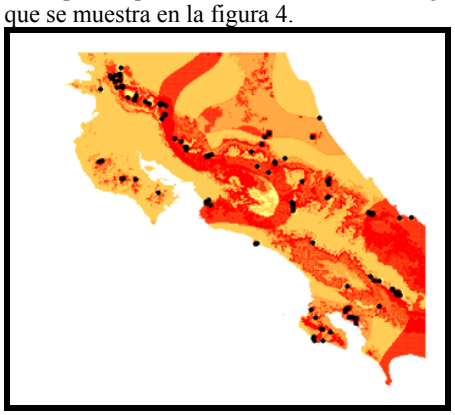


Figura 4. Mapa de distribución potencial de *Anticla antica*.

5. CONCLUSIONES

En el presente trabajo se presentó un enfoque para la generación de mapas de distribución potencial de especies basado en minería de datos espaciales y sistemas de información geográficos. Se mostró como las técnicas de minería de reglas de asociación pueden ser combinadas con técnicas de clasificación para aprovechar las características particulares de este problema, entre las que destaca el hecho de tener un número reducido de consecuentes en las reglas generadas y un objetivo de predicción bien definido el cual es presencia o ausencia de la especie. También se mostró como el uso del concepto de autocorrelación espacial puede mejorar las predicciones.

El prototipo implementado permitió obtener predicciones con altos niveles de precisión y significatividad estadística. Además, los tiempos de ejecución fueron relativamente bajos, no se requirió de ningún equipo sofisticado y las herramientas *Open Source* que se utilizaron llenaron las necesidades satisfactoriamente.

Un aspecto interesante para explorar en futuras investigaciones es el rendimiento de otras técnicas de minería de datos, tales como clasificación pura o segmentación, en este mismo problema. El mayor reto en este sentido seguirá siendo el buscar mecanismos adecuados para integrar la autocorrelación espacial.

Agradecimientos

Los autores desean agradecer al Instituto Nacional de Biodiversidad (INBio) de Costa Rica las facilidades otorgadas para utilizar en esta investigación sus datos de recolecciones biológicas, así como la disponibilidad de su personal para evacuar las preguntas relacionadas con el tema y por apoyar y estimular la realización de este trabajo.

Referencias

[1] Adhikary, J., Han, J., and Koperski, K., Spatial data mining: Progress and challenges. *Database Group Page at Simon Fraser University*, URL: http://db.cs.sfu.ca/GeoMiner/survey/html/survey.html, 1997.

[2] Agrawal, R., Imielinski, T. and Swami, A., Mining Association Rules between Sets of Items in Large Databases, *Proceedings of the ACM SIGMOD international conference on management of data*, Washington DC, USA, 1993.

[3] Cox, B., Moore, P., *Biogeography: an ecological and evolutionary approach*, Blackwell Science, London, 2000.

[4] Chawla, S., Shekhar, S., Wu, W. and Ozesmi, U., Modeling spatial dependencies for mining geospatial data: an introduction, *Geographic data mining and knowledge discovery*, USA, 2001.

[5] Chen, M., Han, J. and Yu, P., Data Mining: An overview from a database perspective, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, Diciembre 1996, pp 866-883.

[6] Cressie, N. A., Statistics for spatial data (revised edition), Wiley, New York, 1993.

[7] Han, J., Koperski, K. and Stefanovic, N., Geominer: a system prototype for spatial data mining, *Proceedings of the ACM SIGMOD international conference on management of data*, 1997.

[8] Kang, I., Kim, T. and Li, K., A spatial data mining method by Delaunay triangulation, *Proceedings of the 5th international workshop on advances in geographic information systems*, 1997.

[9] Kang, I., Kim, T., Li, K. and Son, K., A spatial data mining method by clustering analysis, *Proceedings of the 6th international workshop on advances in geographic information systems*, 1998.

[10] Koperski, K. and Han, J., Discovery of Spatial Association Rules in Geographic Information Databases, *Proc. Fourth International Symposium on Large Spatial Databases*, Maine, 1995, pp. 47-66.

[11] Liu, B., Hsu, W. and Ma, Y., Integrating classification and association rule mining, *KDD*, New York, 1998.

[12] Obando, V. *Biodiversidad en Costa Rica, Estado del conocimiento y gestión*, Editorial INBio, Santo Domingo de Heredia, Costa Rica, 2002.

[13] Piatetsky-Shapiro, G., Frawley, W.J., "Knowledge discovery in databases", AAAI/MIT Press, 1991.

[14] Ram, S., Park, J. and Ball, G., Semantic Model Support for Geographic Information Systems. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 32, No. 5, Mayo 1999, pp. 74-81.

[15] Scott, J.M., Heglund P.J. and Morrison, M., editors, *Predicting species occurrences: Issues of scale and accuracy*. Island Press, Covello, California. 2001.

[16] Segurado, P., Araujo, M. An evaluation of methods for modelling species distributions. *Journal of Biogeography*. Vol. 31. 2004, pp. 1555-1568.

[17] Soberón, J. and Peterson, T. Biodiversity informatics: managing and applying primary biodiversity data. *Philosophical Transactions of the Royal Society of London B*, 359, 2004, pp. 689-698.

[18] Stockwell, D., Overview of Computational Biodiversity Research. Biodiversity Insight Systems. URL: http://biodi.sdsc.edu/Doc/BIS/overview.html. 1997.

[19] Stockwell, D. and Peters, D. The GARP modelling system: problems and solutions to automated spatial prediction. *International Journal of Geographical Information Science*. Vol 13, No. 2. 1999, pp. 143-158.

[20] Tobler, W.R., *Cellular Geography, Philosophy in Geography*, Gale and Odson Eds, Dordrecht, Reidel, 1979.

[21] Wang, W., Yang, J. and Muntz, R., An Approach to Active Spatial Data Mining Based on Statistical Information, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 5, Setiembre/Octubre 2000, pp. 715-728.

Implementación de Primitivas SQL para Reglas de Asociación en una Arquitectura Fuertemente Acoplado

Ricardo Timarán Pereira *
ritimar@udenar.edu.co

Mario Fernando Guerrero*
mfguerrero@hotmail.com

Mario Fernando Díaz*
mferchoz@yahoo.com

Camilo Andrés Cerquera*
feralymph@yahoo.es

and

Stiven Armero*
canonix@gmail.com

*Universidad de Nariño, Departamento de Sistemas,
San Juan de Pasto, Colombia

Abstract

Researchs on Knowledge Discovery in Databases (KDD) was initially oriented toward the definition of new pattern discovery models and development of corresponding algorithms. At present, research has focused on issues related to integrating KDD with database systems, to generate systems and tools for KDD whose architectures can be classified in one of three categories: loosely coupled, middlely coupled and tightly coupled with a Database Management System (DBMS).

In this paper, the process of implementation of SQL primitives for mining association rules on top of a PostgreSQL DBMS is presented and its performance evaluation.

Keywords: Data Base Management Systems, Knowledge Discovery in Databases, Association Task.

Resumen

Las investigaciones en Descubrimiento de Conocimiento en Bases de Datos (DCBD), se centraron inicialmente en definir modelos de descubrimiento de patrones y desarrollar algoritmos para éstos. Investigaciones posteriores se han focalizado en el problema de integrar DCBD con sistemas de bases de datos, produciendo como resultado el desarrollo de sistemas y herramientas de Descubrimiento de Conocimiento cuyas arquitecturas se pueden clasificar en tres categorías: débilmente, medianamente y fuertemente acopladas con un Sistema de Gestión de Bases de Datos (SGBD).

En este artículo se presenta el proceso de implementación de primitivas SQL para descubrir reglas de Asociación al interior del SGBD PostgreSQL y la evaluación de su rendimiento.

Palabras claves: Sistema Gestor de Bases de Datos, Descubrimiento de Conocimiento en Bases de Datos, Tarea de Asociación.

1. Introducción

El Descubrimiento de Conocimiento en Bases de Datos (DCBD) es básicamente un proceso automático en el que se combinan descubrimiento y análisis. El proceso consiste en extraer patrones en forma de reglas o funciones, a partir de los datos, para que el usuario los analice. Esta tarea implica generalmente preprocesar los datos, hacer minería de datos (*Data Mining*) y presentar resultados. DCBD se puede aplicar en diferentes dominios por ejemplo, para determinar perfiles de clientes fraudulentos (evasión de impuestos), para descubrir relaciones implícitas existentes entre síntomas y enfermedades, entre características técnicas y diagnóstico del estado de equipos y máquinas, para determinar patrones de compra de los clientes en sus canastas de mercado, entre otras [26].

Las investigaciones en DCBD, se centraron inicialmente en definir nuevas operaciones de descubrimiento de patrones y desarrollar algoritmos para éstas [1], [5], [20], [6]. Investigaciones posteriores [7], [4], [22], [3], [12], [14], [16], [17] se han enfocado en el problema de integrar DCBD con Sistemas Gestores de Bases de Datos (SGBD), haciendo de ésta un área activa de investigación. Los enfoques de integración de DCBD y SGBD reportados en la literatura se pueden ubicar en uno de tres tipos de arquitectura: sistemas débilmente acoplados, medianamente acoplados y fuertemente acoplados [25].

Una arquitectura es débilmente acoplada cuando los algoritmos de Minería de Datos y demás componentes se encuentran en una capa externa al SGBD, por fuera del núcleo y su integración con éste se hace a partir de una interfaz [25]. Una arquitectura es medianamente acoplada cuando ciertas tareas y algoritmos de descubrimiento de patrones se encuentran formando parte del SGBD mediante procedimientos almacenados o funciones definidas por el usuario [25]. Una arquitectura es fuertemente acoplada cuando la totalidad de las tareas y algoritmos de descubrimiento de patrones forman parte del SGBD como una operación primitiva, dotándolo de las capacidades de descubrimiento de conocimiento y posibilitándolo para desarrollar aplicaciones de este tipo [25]. Debido a que todos los algoritmos son ejecutados en el mismo espacio de direccionamiento que los datos en el SGBD, la ventaja potencial de este enfoque es que resuelve los problemas de escalabilidad y rendimiento de las otras arquitecturas.

Hay propuestas de investigación que discuten la manera como tales sistemas pueden ser implementados y las extensiones del lenguaje SQL necesarias para soportar estas operaciones de descubrimiento de patrones: DMQL [13], M-SQL [15], MINE RULE [17], [18], [19], NonStop SQL/MX [8], [9], Count By Group [11], FilterPartition [23] y Associator Range [27].

En este artículo se implementan los nuevos operadores algebraicos *Associator* y *Equikeep* y las nuevas primitivas SQL *Associator Range* y *EquiKeep On* propuestas en [27] [28] para integrar la tarea de asociación al interior de un SGBD. El resultado es un sistema DCBD fuertemente acoplado con el SGBD PostgreSQL [24] para soportar el descubrimiento de reglas de Asociación en bases de datos.

El resto del artículo se organiza de la siguiente manera: en la sección 2 se presenta los conceptos preliminares sobre reglas de asociación y procesamiento de consultas. En la sección 3 se describe el método tres-pasos para acoplar fuertemente a Asociación con un SGBD. En la sección 4 se describe la manera como se implementan las nuevas primitivas SQL para Asociación al interior del SGBD PostgreSQL. En la sección 5 se reporta el resultado de la evaluación del rendimiento de esta implementación. Finalmente, en la sección 6 se presentan las conclusiones.

2. Conceptos Preliminares

2.1 Reglas de Asociación

El problema de encontrar reglas de asociación fue formulado por Agrawal et al [1] [5] y a menudo se referencia como el problema de canasta de mercado (market-basket). En este problema se da un conjunto de items y una colección de transacciones que son subconjuntos (canastas) de estos items. La tarea es encontrar relaciones entre la presencia de varios items en esas canastas.

Formalmente, sea I= {i1, i2, ...,im} un conjunto de literales, llamados items. Sea D un conjunto de transacciones, donde cada transacción T es un conjunto de items tal que $T \subseteq I$. Cada transacción se asocia con un identificador, llamado TID. Sea X un conjunto de items. Se dice que una transacción T contiene a X si y solo si $X \subseteq T$. Una regla de asociación es una implicación de la forma $X \Rightarrow Y$, donde X y Y son conjuntos de items tal que $X \subset I$, $Y \subset I$ y $X \cap Y = \phi$. El significado intuitivo de tal regla es que las transacciones de la base de datos que contienen X tienden a contener Y. La regla $X \Rightarrow Y$ se cumple en el conjunto de transacciones D con una confianza c si el c% de las transacciones en D que contienen X también contienen Y. La regla $X \Rightarrow Y$ tiene un soporte s en el conjunto de transacciones D si el s% de las transacciones en D contienen $X \cup Y$.

Un ejemplo de una regla de asociación es: "el 30% de las transacciones que contienen cerveza también contienen pañales; el 2% de todas las transacciones contienen ambos items" [2]. Aquí el 30% es la confianza de la regla y el 2%, el soporte de la regla.

La confianza denota la fuerza de la implicación y el soporte indica la frecuencia de ocurrencia de los patrones en la regla. Las reglas con una confianza alta y soporte fuerte son referidas como reglas fuertes (*strong rules*) [1]. El problema de encontrar reglas de asociación se descompone en los siguientes pasos:

- Descubrir los itemsets frecuentes, i.e., el conjunto de itemsets que tienen el soporte de transacciones por encima de un predeterminado soporte *s* mínimo.

- Usar los itemsets frecuente para generar las reglas de asociación para la base de datos.

Después de que los itemsets frecuentes son identificados, las correspondientes reglas de asociación se pueden derivar de una manera directa.

2.2 Procesamiento de una consulta en el SGBD PostgreSQL

El procesamiento de una consulta en el SGBD PostgreSQL incluye las siguientes etapas (Figura 1):

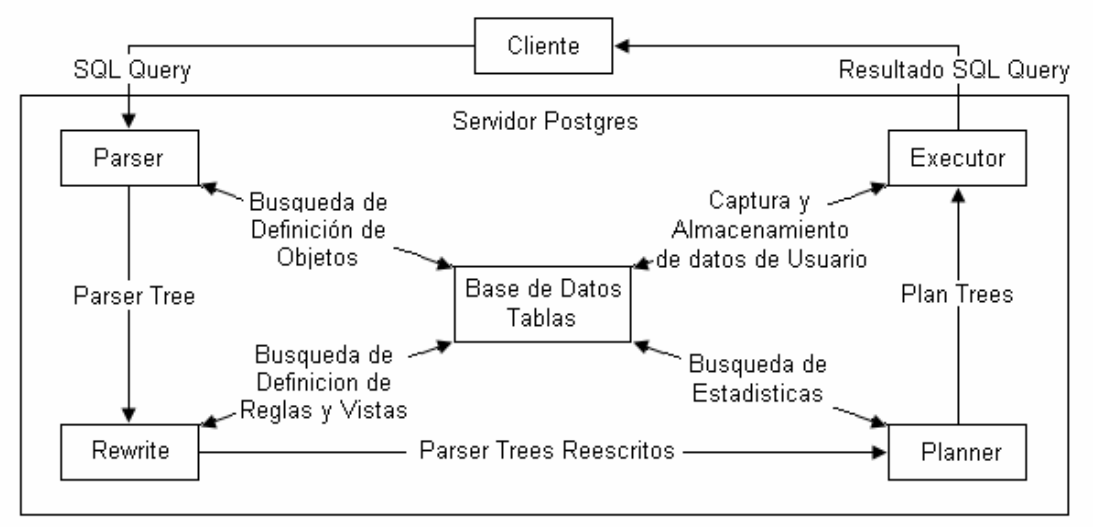


Figura 1. Procesamiento de una consulta en Postgres

- Se establece una conexión desde un programa aplicación al servidor Postgres. A través de esta conexión, el programa aplicación transmite las consultas y recibe los resultados del Servidor Postgres. Las consultas son transmitidas como una cadena de texto plano.
- Se realiza un análisis léxico, sintáctico y semántico a la consulta transmitida. Entre las comprobaciones que se realizan, se verifica la existencia de las relaciones y atributos indicados en la consulta. Si la consulta no presenta errores se crea un árbol de consulta llama *query tree*. Esta etapa es conocida como *etapa Parser.*
- En la siguiente etapa se toma la estructura *query tree*, y se modifican las Reglas o Vistas (VIEW) que se puedan aplicar dentro de la estructura *query tree*. Esta transformación la realiza el sistema *rewrite*, sistema de reescritura. Siempre que se presenta una Vista en la consulta el sistema *rewrite*, transforma la consulta del usuario en una consulta que acceda a las tablas base, dadas en la definición de la vista inicial.
- En la etapa de optimización, se toma la estructura *query tree*, resultado de la etapa *rewrite*, y se crean todas las posibles rutas de acceso que conduzcan al mismo resultado. Se selecciona aquella ruta de acceso que tenga el menor costo de ejecución y se crea el plan de la consulta, dentro de la estructura *query plan*.
- El encargado de ejecutar la estructura *query plan* y de recuperar las tuplas indicadas en la consulta es el *executor*. En esta etapa, se toma la estructura *query plan* y se ejecuta recursivamente. Se buscan relaciones, se realizan ordenamientos y Joins, se evalúan cualificaciones, se hace uso del sistema de almacenamiento y se recuperan las tuplas en la forma representada en el plan.

3. Método Tres-Pasos para la Integración Fuerte de una Tarea de Minería de Datos

En [27] se propone un método que permite descubrir conocimiento eficientemente al interior de un SGBD y facilita la integración fuerte de una tarea de minería de datos con un SGBD. El método esta compuesto por tres pasos: primero, se extiende el álgebra relacional con nuevos operadores algebraicos que faciliten los procesos computacionalmente más costosos de la tarea de minería de datos; segundo, se extiende el lenguaje SQL con nuevas primitivas expresadas en la cláusula SQL SELECT, las cuales implementan los nuevos operadores algebraicos y tercero, se unifica estas primitivas en un nuevo operador SQL que se expresa en una nueva cláusula SQL y que obtiene el resultado de la tarea de minería de datos.

Específicamente para Asociación, en [27][28] se propone:

3.1 Operadores algebraicos para Asociación

Extender el álgebra relacional [10] con los siguientes operadores algebraicos:

- *Associator (α)*. Este operador genera, por cada tupla de la relación *R*, todos sus posibles subconjuntos (itemsets) de diferente tamaño, en una sola pasada.
- *EquiKeep (χ)*. Este operador restringe los valores de los atributos de cada una de las tuplas de la relación *R* a únicamente los valores de los atributos que satisfacen una expresión lógica.

3.2 Primitivas SQL para Asociación

Extender el lenguaje SQL con las siguientes primitivas:

- *Primitiva Associator Range*. Esta primitiva implementa el operador algebraico *Associator* [28] en la cláusula SQL SELECT. Associator permite obtener por cada tupla de una tabla, todos los posibles subconjuntos desde un tamaño inicial hasta un tamaño final determinado por la cláusula RANGE. Dentro de la cláusula SELECT, *Associator* tiene la siguiente sintaxis:

```
SELECT <ListaAtributosTablaDatos> [INTO <NombreTablaAssociator>]
FROM  <NombreTablaDatos>
WHERE <CláusulaWhere>
ASSOCIATOR  RANGE <numero1> UNTIL <numero2>
GROUP BY <ListaAtributosTablaDatos>
```

- *Primitiva Equikeep On*. Esta primitiva implementa el operador algebraico *EquiKeep* [27] en la cláusula SQL SELECT. *EquiKeep on* conserva en cada registro de una tabla los valores de los atributos que cumplen una condición determinada. El resto de valores de los atributos se hacen nulos. Dentro de la cláusula SELECT, *EquiKeep on* tiene la siguiente sintaxis:

```
SELECT <ListaAtributosTablaDatos> [INTO <NombreTablaEquiKeep>]
FROM  <NombreTablaDatos>
WHERE <CláusulaWhere>
EQUIKEEP ON  <CondiciónValoresAtributos >
```

3.3 Operador unificado Describe Association Rules

Unificar las primitivas anteriores en un solo operador SQL denominado Describe Association Rules. Este operador genera todas las reglas de asociación de una longitud determinada, que cumplen con un soporte y una confianza mayor o igual que unos determinados umbrales especificados por el usuario. *Describe Associator Rules* tiene la siguiente sintaxis:

```
DESCRIBE [UNIDIMENSIONAL][MULTIDIMENSIONAL] ASSOCIATION RULES
FROM  <NombreTablaItemsetsFrecuentes> [INTO <TablaReglasAsociación>]
WITH CONFIDENCE  <valor1>
LENGTH <valor2 >
[AS  <SubconsultaItemsetsFrecuentes>]
```

4. Aspectos de la Implementación de las Primitivas SQL para Asociación en el interior de Postgres

La implementación de las nuevas primitivas SQL para dotar al SGBD PostgreSQL [34] de las capacidades de descubrir reglas de Asociación implicó realizar las siguientes modificaciones en algunas de sus funciones y estructuras de sus componentes:

4.1 Parser

El analizador léxico no se altera, al no haber nuevos símbolos. Se modifica el analizador sintáctico con nuevo código para lograr que el compilador de Postgres reconozca la sintaxis de las nuevas primitivas con los siguientes procedimientos:

Con el fin de introducir las nuevas palabras reservadas *Associator*, *Range*, *Equikeep*, *On*, se modifica el archivo *…/src/backend/parser/keywords.c* donde las palabras reservadas se listan en una estructura especial, manteniendo un estricto orden alfabético.

En el archivo *…/src/backend/parser/gram.y* se adiciona las nuevas reglas de producción siguiendo la estructura lógica de este archivo especial de gramática para hacer funcional las nuevas producciones sin alterar las existentes. En el archivo *…/src/include/nodes/parsenodes.h se* modifica las estructuras *SelectStm*t, y *Query* para introducir nuevos enlaces a nodos o estructuras de uso exclusivo de las nuevas primitivas.

Al obtener las estructuras con la forma que se necesita, se procede a modificar el archivo *…/src/backend/parser/analize.c* en el cual desencadena el análisis semántico. En la función *transformSelectStmt* se realiza la llamada a las funciones *transformAssociatorClause*, *transformEquikeepClause* implementadas físicamente en el archivo *…/src/backend/parser/parseclause.c* para que a partir de la estructura base *Selectstmt*, se genere la estructura definitiva *Query* que toma como entrada el *Rewriter*.

La nueva estructura del *Query Tree* se muestra en la figura 2.

4.2 Rewriter

Puesto que el proyecto no contempla la implementación de las primitivas Data Mining como parte de la definición de vistas, ni reglas de transformación a partir de éstas, no se realizó ninguna modificación a los módulos que pertenecen al rewriter.

4.3 Planner/Optimizer

El Planner, tiene por objetivo la creación de un nuevo plan de ejecución. Se realizaron los siguientes cambios a su estructura:

Primero se adiciona en el archivo *…/src/include/nodes/nodes.h* el tag (etiqueta) para reconocimiento de los nuevos nodos (T*_Associator*, *T_Equikeep*) en su respectiva sección.

Segundo, en el archivo *…/src/include/nodes/plannodes.h* se incluyen los nodos *Associato*r y *Equikeep*. Estos nuevos nodos deben incluir en su estructura un nodo de estado del cual se hará uso en la etapa de ejecución.

Tercero, se incluye el prototipo de las funciones *make_Associator* y *make_Equikeep* en el archivo *…/src/include/optimizer/planmain.h* y se implementa físicamente en el archivo *…/src/backend/optimizer/plan/createplan.c* en las cuales a partir del *path* seleccionado, se crea el plan con su respectivo costo.

Cuarto, se incluye la llamada de éstas funciones desde la función *grouping_planner* en el archivo *…/src/backend/optimizar/plan/planner.c.*

El plan de ejecución para las primitivas de asociación se muestra en la figura 3.

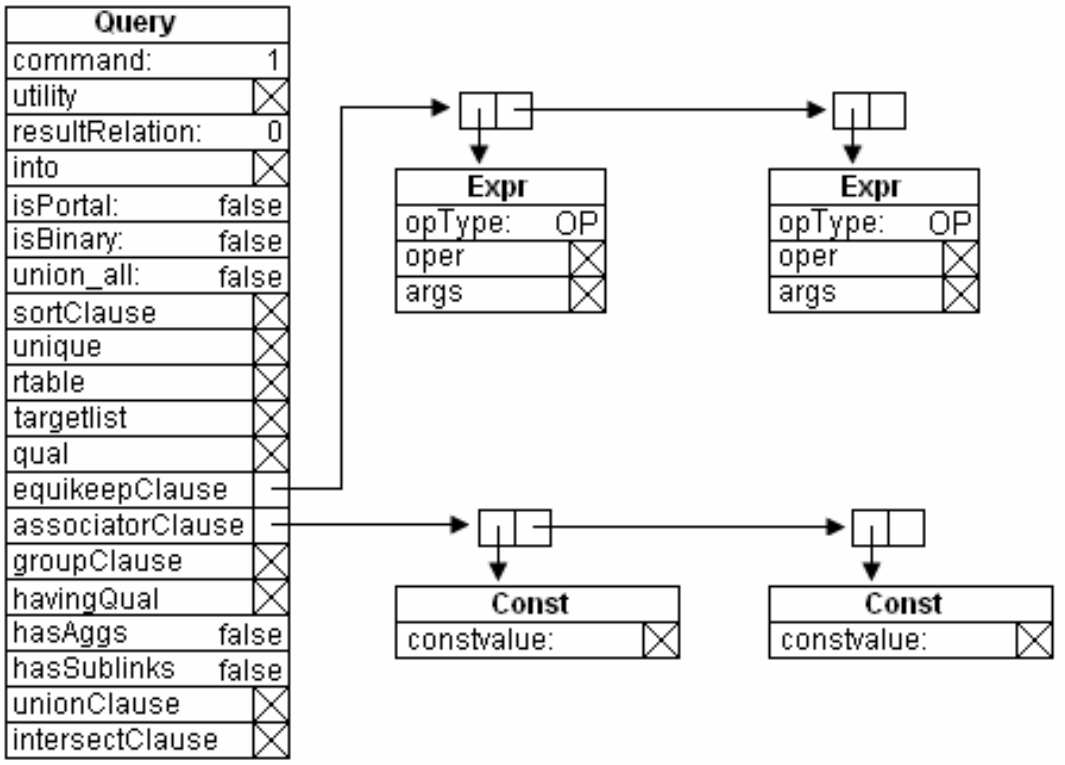


Figura 2. Nueva estructura Query tree

4.4 Executor

Se crean los archivos *nodeAssociator.c, nodeAssociator.h*, *nodeEquikeep.c* y *nodeEquikeep.h* en los cuales se define la interfaz de los respectivos nodos para la ejecución particular de estos planes. Los archivos creados con extensión *.c* se ubican en *…/src/backend/executor/* y los archivos con extensión .h en*…/src/include/executor/.* Cada archivo *.c* contiene tres funciones básicas que corresponden a las fases del executor:

- Inicializar. La función de inicialización debe configurar el estado de ejecución, la dirección de búsqueda, conjunto de relaciones que intervienen y bloques de memoria disponibles.

- Ejecutar. La función de ejecución realiza el procesamiento real del plan generado en la etapa anterior y retorna las tuplas que satisfacen el plan.

- Finalizar. Esta función se encarga de liberar todos los recursos que se reservaron en la ejecución del plan

Para la primitiva Associator, estas rutinas básicas se muestran en la figura 4.

5. Evaluación del Rendimiento

Para evaluar el rendimiento de las primitivas implementadas, se utilizó un equipo Pentium IV a 2.8 Ghz, con 512 MB de RAM y bajo sistema operativo Red Hat Linux 9. La versión de PostgreSQL fue la 7.3.4.

El conjunto de datos utilizados en las pruebas pertenecen a las transacciones de un supermercado de una de las Cajas de Compensación más importantes del departamento de Nariño durante un periodo determinado. El tamaño de la primera muestra es de 517.956 transacciones, con un promedio de cinco productos por transacción y pertenecientes a 10.757 diferentes productos; la segunda muestra es de 1.039.906 transacciones, con un promedio de diez productos por transacción y que pertenecen a la misma cantidad de productos de la muestra anterior.

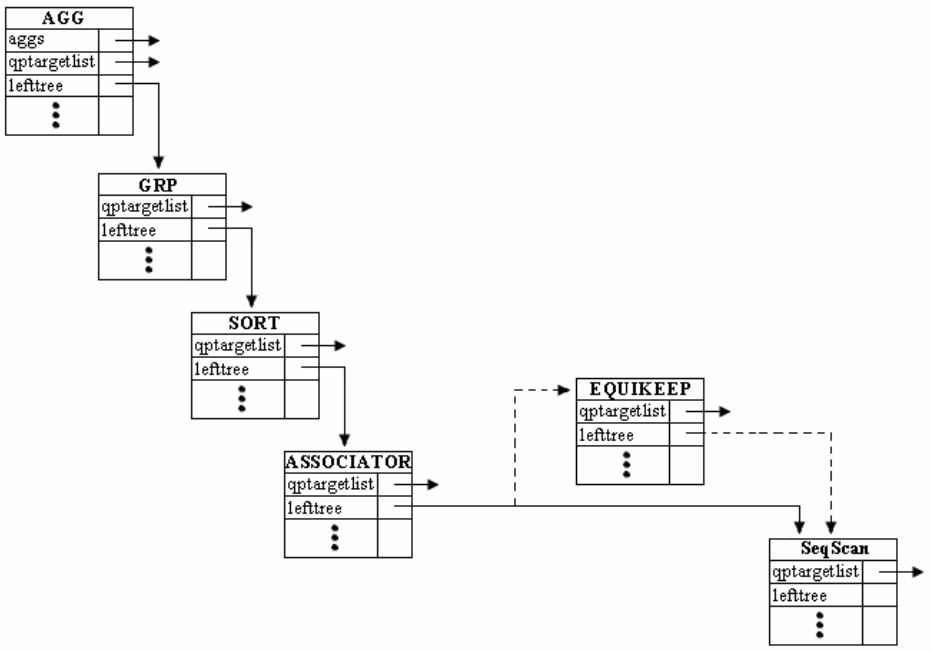


Figura 3. Plan de ejecución de las Primitivas de Asociación

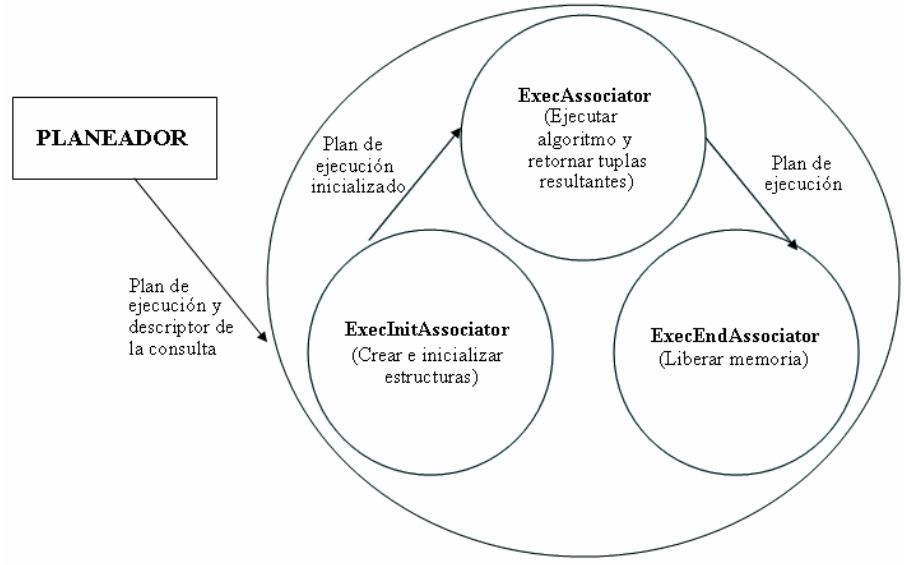


Figura 4. Rutinas de manipulación para Associator

Para su análisis, las dos primitivas SQL de Asociación: *Associator Range* y *EquiKeep* se utilizaron en conjunto con el fin de reducir significativamente el tiempo de generación de los itemset frecuentes. *Equikeep* permite conservar únicamente los items frecuentes en cada transacción, mientras que *Associator Range* asocia éstos items par generar los itemsets frecuentes del tamaño indicado [27].

Se realizaron varias tareas de preprocesamiento previas a la aplicación de las primitivas de Asociación sobre el conjunto de datos. Se eliminaron productos repetidos en una misma transacción y se transformó el conjunto de datos del modelo simple al modelo multicolumna [21], con el cual operan las primitivas *Associator Range* y *EquiKeep*. El resultado final se almacenó en dos tablas Postgres llamadas T5I3D83K y T10I4D145K [5], dónde en la primera tabla el número de

ítems promedio por transaccion es 5, con tamaño de 3 itemsets frecuentes potenciales y un total de 83.720 registros válidos (83K) y para la segunda el tamaño promedio de ítems por transacción es 10, con tamaño de 4 itemsets frecuentes potenciales y un total de 144.954 registros válidos (144K).

La metodología utilizada consistió en aplicar las nuevas primitivas de asociación sobre las bases de datos, comparando los tiempos de respuesta y la cantidad de registros resultantes, para diferentes soportes mínimos.

La tabla 1 muestra los tiempos de ejecución y el número de large itemsets obtenidos al aplicar las primitivas de asociación con diferentes soportes sobre la base de datos T5I3D83K, como también el número de reglas generadas con su respectivo tamaño y confianza. Los tiempos de ejecución excluyen el preprocesamiento; Para T5I3D83K se utilizaron porcentajes de soporte entre 0.6% y 0.06% ya que para porcentajes menores se obtienen una gran cantidad de reglas no válidas, lo que conlleva a un análisis posterior poco significativo.

Soporte (%)	Confianza (%)	No. Reglas	Tamaño Reglas	Tiempo (Seg.)	Número de large Itemsets		
					Total	Tam 2	Tam 3
0.6	20	0	2	7	5	5	0
0.3	20	0	2	28	10	10	0
0.1	20	31	3	206	54	41	13
0.06	20	62	3	423	137	106	31

Tabla 1. Resultados obtenidos para T5I3D83K al variar el soporte mínimo

Los resultados de las pruebas para T5I3D83K se resumen en las Figuras 5 y 6. En la Figura 5 se muestra el tiempo de ejecución de las primitivas de asociación y el número total de itemsets frecuentes. La gráfica indica que para un soporte de 0.6% resultaron 5 2-itemsets frecuentes en un tiempo de 7 segundos; para 0.3%, 10 2-itemsets frecuentes en 28 segundos; para 0.1%, 54 itemsets frecuentes de los cuales 41 corresponden a tamaño 2 y 13 a tamaño 3, en 206 segundos; así mismo para 0.06%, resultaron 137 itemsets frecuentes de los cuales 106 corresponden a tamaño 2 y 31 a tamaño 3, en 423 segundos.

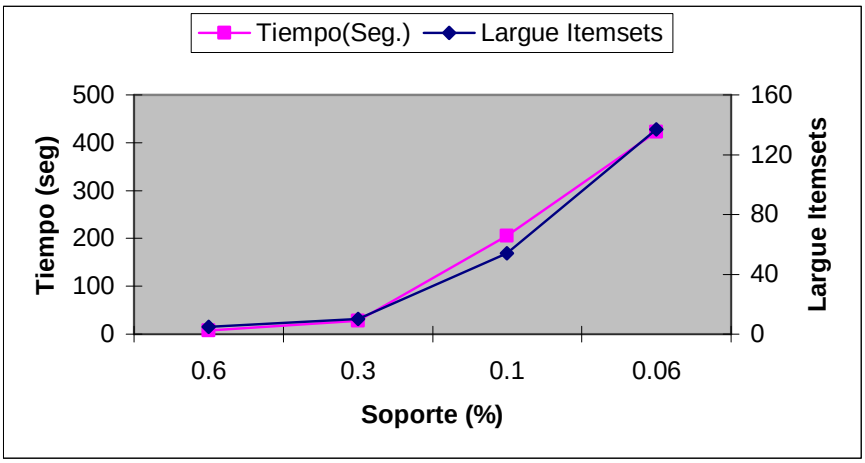


Figura 5. Resultados obtenidos para T5I3D83K

En la Figura 6 se presentan algunas de las reglas más representativas obtenidas por medio del operador SQL *Describe Associator Rules*, con los parámetros antes mencionados, donde la primera columna representan el número de la regla generada y la segunda columna el antecedente y consecuente de la regla. Por ejemplo la regla número 113, indica que "el 70% de los clientes que compraron Fríjol Cargamento rojo y Maizena Top también compraron Lenteja"; o la regla número 58 "el 21% de los clientes que compraron Fríjol Lima y Arveja verde también compraron Garbanzo".

Nro. Regla	Regla	
	Antecedentes	Consecuentes
113	600012 ^ 600023	600015
58	600002 ^ 600017	600019

Representa:

Nro. Regla	Regla	
	Antecedentes	Consecuentes
113	Fríjol Cargamanto Rojo ^ Maizena Top	Lenteja
58	Fríjol Lima ^ Arveja Verde	Garbanzo

Figura 6. Reglas Representativas para T5I3D83K

Para la base de datos T10I4D145K, se realizó pruebas con soportes entre 0.5% y 0.05% y confianza del 30%. Los resultados se muestran en la tabla 2.

Soporte (%)	Confianza (%)	No. Reglas	Tamaño Reglas	Tiempo (Seg.)	Número de large Itemsets			
					Total	Tam 2	Tam 3	Tam 4
0.5	30	0	0	25	11	10	1	0
0.3	30	0	0	124	28	21	7	0
0.1	30	97	4	730	160	119	36	5
0.05	30	238	4	1653	510	380	112	18

Tabla 2. Resultados obtenidos para T10I4D145K al variar el soporte mínimo

Los resultados de analizar los datos de la prueba anterior se muestran en las figuras 7 y 8. La Figura 7 indica que para un soporte de 0.5% resultaron 11 itemsets frecuentes en un tiempo de 25 segundos; para 0.3%, 28 itemsets frecuentes en 124 segundos; para 0.1%, 160 itemsets frecuentes en 730 segundos; así mismo para 0.05%, resultaron 510 itemsets frecuentes en 1653 segundos. De acuerdo a esto, se considera que, además del número de itemsets frecuentes y de los tamaños de los itemsets, el valor del soporte y la cantidad de transacciones son factores importantes que determinan el tiempo de ejecución.

Conservando el formato de la figura 6, se presentan algunas de las reglas más representativas obtenidas por medio del operador SQL *Describe Associator Rules* en la figura 8, donde la regla número 118, representa que "el 30% de los clientes que compraron Fríjol Lima y Panela también compraron Lenteja y Sal Refisal"; o la regla número 210 "el 72% de los clientes que compraron Arveja Verde, y Panela también compraron Lenteja".

De acuerdo al análisis total de las reglas válidas obtenidas de las 2 muestras (93 reglas para T5I3D83K, 335 reglas para T10I4D145K), la gran mayoría incluye 8 productos básicos (fríjol lima, lenteja, arveja verde, garbanzo, fríjol cargamento rojo, maizena top, panela y sal refisal), el resto se distribuye uniformemente entre los demás productos. Esto indica la preferencia de compra de los clientes por los productos de primer orden de la canasta familiar, a pesar de la gran cantidad de artículos (10.757) registrados en las bases de datos.

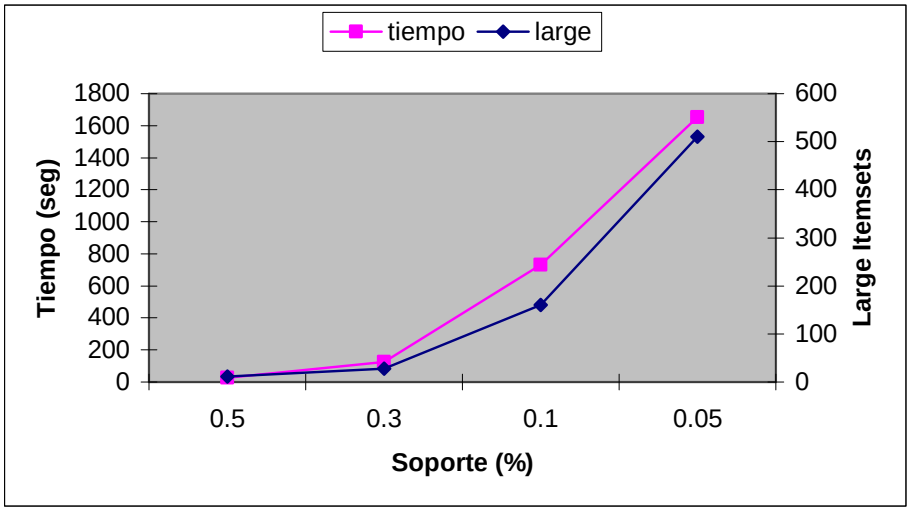


Figura 7. Resultados obtenidos para T10I4D145K

Nro. Regla	Regla	
	Antecedentes	Consecuentes
118	600002 ∧ 600030	600015 ∧ 603005
210	600017 ∧ 600030 ∧ 603005	600015

Figura 8. Reglas Representativas para T10I4D145K

6. Conclusiones

Se han presentado los primeros resultados de la implementación de las nuevas primitivas SQL propuestas en [27][28] para descubrir reglas de asociación al interior de un SGBD. Estos resultados convierten a PostgreSQL en uno de los primeros SGBD con la capacidad de compilar y ejecutar una consulta que involucre descubrimiento de conocimiento. Con ello, PostgreSQL es capaz de soportar consultas de minería de datos "ad-hoc", i.e, consultas flexibles e interactivas ejecutadas sobre un conjunto de datos con el fin de obtener reglas de asociación con diferente soporte y confianza.

A partir de las pruebas realizadas se deduce que el tiempo de ejecución de los operadores de asociación crece linealmente con respecto al número de transacciones de la tabla y exponencialmente con el número de atributos. La primitiva *Equikeep* hace más eficiente la generación de itemsets frecuentes cuando se trabaja con valores altos de soporte y disminuye en eficiencia cuando los valores de soporte son menores debido a que aumenta el número de ítems tamaño uno (1) que hacen parte de la expresión lógica que se deben evaluar.

En el futuro, se analizará el rendimiento de esta arquitectura con respecto a una débilmente acoplada para demostrar su eficiencia. Se continuará con la implementación de otras nuevas primitivas SQL que se han propuesto para Asociación y Clasificación. Se dotará al optimizador de PostgreSQL con nuevas estrategias de transformación y costos que se han definido para los operadores algebraicos relacionales que soportan estas primitivas.

Referencias

[1] Agrawal R., Imielinski T., Swami A., Mining Association Rules between Sets of Items in Large Databases, ACM SIGMOD, Washington DC, USA, 1993.

[2] Agrawal R., Mehta M., Safer J., Srikant R., The Quest Data Mining System, 2º Conference KDD y Data Mining, Porland, Oregon, USA, 1996.

[3] Agrawal R., Mehta M., Shafer J., Srikant R., Arning A., Bollinger T., The Quest Data Mining System, 2º Conference KDD y Data Mining, Portland, Oregon, 1996.

[4] Agrawal R., Shim K., Developing Tightly-Coupled Data Mining Applications on a Relational Database System, The Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996.

[5] Agrawal R., Srikant R., Fast Algorithms for Mining Association Rules, VLDB Conference, Santiago, Chile, 1994.

[6] Brin S., Motwani R., Ullman J., Tsur S., Dynamic Itemset Counting and Implication Rules for Market Basket Data, ACM SIGMOD, USA, 1997.

[7] Chaudhuri S., Data Mining and Database Systems: Where is the Intersection?, Bulletin of the Technical Committee on Data Engineering, Vol. 21 No. 1, Marzo, 1998.

[8] Clear, J., Dunn, D., Harvey, B., Heytens, M., Lohman, P., Mehta, A., Melton, M., Rohrberg, L., Savasere, A., Wehrmeister, R., Xu, M., NonStop SQL/MX Primitives for Knowledge Discovery, KDD-99, San Diego, USA, 1999.

[9] Clear, J., Dunn, D., Harvey, B., Heytens, M., Lohman, P., Mehta, A., Melton, M., Rohrberg, L., Savasere, A., Wehrmeister, R., Xu, M., Large Scale Knowledge Discovery Using NonStop SQL/MX, Technical Report, Compaq Computer Corporation, Tandem Division, 1999.

[10] Codd, E., F., A Relational Model of Data for Large Shared Data Banks, CACM, Vol. 13, No. 6, June, 1970.

[11] Freitas, A.A., Generic, Set-oriented Primitives to Support Data-parallel Knowledge Discovery in Relational Database Systems, Ph.D. diss., http://cswww.essex.ac.uk/SystemsArchitecture/DataMining/alex/thesis.html, 1997.

[12] Han J., Fu Y., Wang W., Chiang J., Koperski K., Li D., Lu Y., Rajan A., Stefanovic N., Xia B., Zaiane O., DBMiner: A System for Mining Knowledge in Large Relational Databases, The second International Conference on Knowledge Discovery & Data Mining, Portland, Oregon, 1996.

[13] Han J., Fu Y., Wang W., Koperski K., Zaiane O., DMQL: A Data Mining Query Language for Relational Databases, SIGMOD 96 Workshop, On research issues on Data Mining and Knowledge Discovery DMKD 96, Montreal, Canada, 1996.

[14] Imielnski T., Mannila, H., A Database Perspective on Knowledge Discovery, Communications of the ACM, Vol 39, No. 11, November, 1996.

[15] Imielinski T., Virmani A., Abdulghani A., Data Mine: Application Programming Interface and Query Language for database Mining, 2º Conference KDD y Data Mining, Portland, Oregon, 1996.

[16] Matheus C., Chang P., Piatetsky-Shapiro G., Systems for Knowledge Discovery in Databases, IEEE Transactions on Knowledge and Data Engineering, Vol 5, No 6, 1993.

[17] Meo R., Psaila G., Ceri S., A New SQL-like Operator for Mining Association Rules, VLDB Conference, Bombay, India, 1996.

[18] Meo R., Psaila G., Ceri S., An Extension to SQL for Mining Association Rules, Data Mining and Knowledge Discovery, Kluwer Academic Publishers, Vol 2, pp .195-224, Boston, 1998.

[19] Meo R., Psaila G., Ceri S., A Tightly-Coupled Architecture for Data Mining, 14th. International Conference on Data Engineering ICDE98, 1998.

[20] Park J.S., Chen M., Yu P., An Effective Hash-Based Algorithm for Mining Association Rules, ACM SIGMOD, San Jose, Ca USA, 1995.

[21] Rajamani, K., Cox, A., Iyer, B., Chadha, A., Efficient Mining for Association Rules with Relational Database Systems, International Database Engineering and Aplication Symposium, p. 148-155, 1999.

[22] Sarawagi, S., Thomas S., Agrawal R., Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications, Data Mining and Knowledge Discovery, Kluwer Academic Publishers, Vol 4, 2000.

[23] Sattler, K., Dunemann, O., SQL Database primitives for Decision Tree Classifiers, CIKM, Atlanta, Georgia, USA, 2001.

[24] Stonebraker, M., Rowe, L., A., The Design of POSTGRES, Proceedings of the ACM-SIGMOD Conference, Washington D.C., 1986.
[25] Timarán, R., Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de bases de datos: un Estado del Arte, en revista Ingeniería y Competitividad, Universidad del Valle, Volumen 3, No. 2, Cali, diciembre de 2001.

[26] Timarán, R., Descubrimiento de conocimiento en Bases de Datos: Una visión general, en memorias del Primer Congreso Nacional de Investigación y Tecnología en Ingeniería de Sistemas, Universidad del Quindío, Armenia, octubre de 2002.

[27] Timarán, R., Millán, M., Machuca,F., New Algebraic Operators and SQL Primitives for Mining Association Rules, in proceedings of the IASTED International Conference on Neural Networks and Computational Intelligence (NCI 2003), International Association of Science and Technology for Development, Cancun, Mexico, mayo 2003.

[28] Timarán, R., Millán, M., EquipAsso: An Algorithm based on New Relational Algebraic Operators for Association Rules Discovery, in proceedings of the IASTED International Conference on Computational Intelligence (CI 2005), International Association of Science and Technology for Development, Calgary, Canadá, julio 2005.

Perspectivas de Conocimiento en Grids de Datos

Luis Eliécer Cadenas, Emilio Hernández
Universidad Simón Bolívar, Departamento de Computación y T. I.,
Apartado 89000, Caracas 1080-A, Venezuela
{leliecer,emilio}@ldc.usb.ve

Abstract

In this paper a methodology for accesing scientific data repositories on data grids is proposed. This methodology is based on ontology specification and knowledge representation. The concept of *Knowledge Perspective* is introduced, as the action of applying particular scientific conjectures or theories to the interpretation of experimental data and information. Data grid environments provide high levels of security and virtualization, which allow the users to create new data services on the data server side. These new services are based on the user's knowledge perspective. An implementation of this concept is presented, on a Globus-enabled Java execution platform.

Keywords: Data Grids, Semantid Grids, Ontologies

Resumen

En este artículo se presenta una metodología para acceder a fuentes de datos contenidas en grids de datos. Esta metodología se basa en especificación de ontologías y representación de conocimientos. Se introduce el concepto de *Perspectiva de Conocimiento* para denotar la acción de aplicar conjeturas o teorías científicas particulares a la interpretación de datos e información de carácter experimental. Los grids de datos proveen los niveles de seguridad y virtualización necesarios para permitirle a los usuarios crear nuevos servicios de datos en el lado del servidor de datos. Estos servicios nuevos están basados en la perspectiva de conocimiento del usuario. Mostramos la implementación del concepto en una arquitectura que ejecuta bytecode de Java sobre una plataforma Grid basada en Globus.

Palabras claves: Grid de Datos, Grid Semántico, Ontologías

1 Introducción

Buena parte de la producción científica actual se apoya en técnicas de cómputo intensivo. Estas técnicas se usan para procesar, generalmente en forma distribuida, grandes cantidades de datos e información. Esta forma de hacer ciencia se ha denominado e-ciencia (*e-science*). Los ambientes *grid* han aportado una parte importante de estas tecnologías. Sin embargo las herramientas disponibles aún distan de poseer la flexibilidad y capacidad requeridas para desarrollar todo el potencial que ofrece la actividad de e-ciencia. En este artículo proponemos y evaluamos el concepto de *Perspectivas de Conocimiento* como una base para manipular datos científicos e información experimental en el ámbito de los Grids de Datos. En un sentido abstracto nos referimos a *perspectiva de conocimiento* para denotar la acción de aplicar una teoría científica al proceso de interpretar datos e información de carácter experimental. Las teorías científicas son enunciados de carácter universal. Los enunciados básicos de carácter universal que componen cualquier teoría científica se aceptan como resultado de una decisión o un acuerdo, y desde ese punto de vista son convenciones [24]. Al seleccionar el conjunto de enunciados básicos que constituyen un determinado marco teórico, seleccionamos el punto de vista desde el cual interpretaremos los datos. Esta selección caracterizará los eventos que tendrán relevancia en el proceso de contrastación de la teoría.

Desde un punto de vista computacional, estos enunciados universales pueden ser expresados formalmente usando un lenguaje ontológico lo suficientemente expresivo como para definir conceptos, relaciones entre conceptos y aseveraciones lógicas de carácter general. El concepto general que proponemos supone el procesamiento de los datos experimentales usando ontologías descriptivas de un dominio de conocimiento dado.

De esta forma una *perspectiva de conocimiento* incluye datos que han sido procesados de acuerdo a una determinada teoría y que le permiten al investigador procesar en forma sistemática, coherente y completa los datos que ha recolectado y almacenado sobre un determinado fenómeno que quiera analizar.

Este enfoque ofrece varias ventajas al científico experimental. En primer lugar le ofrece una metodología para manipular y procesar los datos que es consistente con la interpretación teórica que tiene del fenómeno sobre el cual desarrolla su investigación. Esta metodología consiste en describir en forma precisa y usando un lenguaje formal, la teoría, las hipótesis experimentales que desea contrastar y una descripción de los atributos de los objetos que deben ser identificados en los datos experimentales. Al expresarlo de esta forma le ofrecemos herramientas que le permiten revisar la coherencia lógica de sus aseveraciones y extraer conclusiones adicionales. Con esta ontología y usando algún método de procesamiento que se ajuste a la naturaleza de los datos y que también puede ser definido por él, el investigador puede procesar los datos y generar índices que le permitan acceder en forma eficiente sólo a aquellos datos que son relevantes a las hipótesis teóricas descritas por él. Al hacerlo de esta forma conseguimos expresar explícitamente la relación entre la teoría, sus conclusiones y los datos experimentales que la corroboran, en una forma que es, al menos en principio, reproducible por cualquier otro investigador.

En segundo lugar le ofrecemos la flexibilidad necesaria para que los mismos datos experimentales puedan ser vistos desde diversos puntos de vista. Estos puntos de vista pueden ser diversos debido a que representan la interpretación de los datos desde teorías distintas, a que están siendo manipulados por científicos que provienen de diversas disciplinas o porque las perspectivas definidas corresponden a partes distintas de un mismo cuerpo teórico.

En tercer lugar, el procesamiento que se realiza al construir una perspectiva sobre una fuente de datos permite su manipulación eficiente. En este campo encontramos con frecuencia fuentes de datos que alcanzan varios terabytes de información y que pronto serán de varios petabytes generados por año [17].Usando una ontología de la forma en que proponemos puede reducirse el tamaño del conjunto de datos que el investigador tiene que acceder. Esta técnica es especialmente útil si este procesamiento se realiza una vez y luego se accede a los datos múltiples veces.

Debido a que la construcción de la perspectiva no implica ningún cambio a los datos originales se protege la calidad y confiabilidad de los datos experimentales sin restarle flexibilidad al investigador. Las perspectivas, como es de suponerse, pueden componerse, de tal forma que pueden haber perspectivas sobre perspectivas lo cual ofrece interesantes posibilidades de reinterpretación y validación.

El aporte fundamental de este trabajo es un modelo computacional epistemológicamente consistente con la naturaleza de la investigación científica y que tiene una base operacional que permite la manipulación eficiente de grandes fuentes de datos distribuidas, especialmente en Grids de datos, ya que éstos ofrecen las herramientas de seguridad necesarias para permitir el acceso a recursos remotos de una manera segura. Hemos hecho una implementación de este modelo en una herramienta llamada SUMA/G [16], un sistema de cómputo distribuido para ejecución de programas Java, implementado sobre varios de los servicios de Globus, con énfasis en la infraestructura de seguridad.

El resto de este artículo está organizado de la siguiente forma. La Sección 2 formaliza el concepto de perspectiva de conocimiento y su relación con la actividad científica. La Sección 3 muestra la arquitectura general para implementar un servicio de perspectivas de conocimiento en ambientes grids. La Sección 4 ofrece una descripción de la arquitectura de SUMA-onto, una implementación del concepto en SUMA/G . La Seccion 5 muestra la aplicación del concepto de perspectivas para mejorar la recuperación de artículo científicos a través del uso de Wordnet. La sección 6 muestra los trabajos relacionados y la sección 7 ofrece nuestras conclusiones y detalla el trabajo futuro.

2 Perspectivas de conocimiento e investigación científica

Introducimos el concepto de *Perspectiva de Conocimiento*, que puede ser caracterizado operativamente a través de una tupla Π de la forma:

$$\Pi = (\Gamma, \Omega, \Lambda)$$

en la cual Γ es el conjunto de objetos de interés (x_i) contenidos en la fuente de datos:

$$\Gamma = \{x_1, x_2, x_3, \ldots x_{n_2}\}$$

sobre los que aplica un conjunto de predicados $P = \{p_1, p_2, p_3, \ldots p_{n_1}\}$, dando lugar a Ω, que es un conjunto de conjuntos Φ_i:

$$\Omega = \{\Phi_1, \Phi_2, \Phi_3, \ldots \Phi_{n_1}\}$$

Cada conjunto Φ_i contiene todas la tuplas de objetos pertenecientes a Γ que satisfacen el predicado p_i. Si el predicado p_i corresponde a una propiedad entonces las tuplas que integran Φ_i son unarias. Si p_i representa una relación, las tuplas que componen Φ_i tendrán el número de objetos relacionados. Cada Φ_i representa un término de la ontología usada para el procesamiento de la perspectiva. Algunos de los conjuntos Φ_i podrían ser el resultado de un proceso de *marcado ontológico* que consiste en aplicar un procesamiento directamente a los datos de Γ. En concreto, este procesamiento verificaría el predicado p_i sobre los datos, y debe ser aportado por el investigador como parte de su perspectiva de conocimiento. Por último, Λ es un conjunto, posiblemente vacío, constituido por aseveraciones A_i. Es necesario que cada aseveración $A_i = W_i(x_1, x_2 \ldots x_{h_i})$ sea una fórmula bien formada de la lógica de predicados, en el sentido de que cada variable esté ligada a un cuantificador (existencial o universal):

$$\Lambda = \{A_1, A_2, A_3 \ldots A_{n_3}\}$$

Como un ejemplo para aclarar las definiciones anteriores podríamos decir que Γ podría ser un repositorio de imágenes astronómicas tomadas en diferentes momentos, Ω un conjunto de conjuntos de estrellas con la misma magnitud aparente y Λ un conjunto de aseveraciones acerca de la formación de supernovas, que implica la variación de la magnitud aparente en una determinada escala de tiempo. El cálculo de las magnitudes aparentes, es decir, la creación del conjunto Ω, se realiza a partir de un marcado ontológico de los elementos de Γ, que puede ser el producto del procesamiento directo de las imágenes o realizarse a partir de un catálogo de estrellas pre-existentes.

Las aseveraciones A_i pueden representar conjeturas o definiciones sobre las relaciones entre objetos en la fuente de datos o entre sus propiedades. Al mismo tiempo pueden ser usadas para derivar nuevos predicados, es decir, nuevos conjuntos Φ_i que pueden ser agregados a los ya existentes en el conjunto Ω para crear un conjunto Ω'. Definimos entonces el cálculo de una perspectiva como la materialización del conjunto Ω' a partir de Ω, después de aplicarle las aseveraciones pertenecientes al conjunto Λ. En este caso las tripletas $(\Gamma, \Omega, \Lambda)$ y $(\Gamma, \Omega', \Lambda)$ contienen la misma información, es decir, representan la misma perspectiva de conocimiento. Para obtener una perspectiva de conocimiento diferente, sobre los mismos datos Γ, se debe cambiar el marcado ontológico de los datos y/o el conjunto Λ, en el sentido de que contenga aseveraciones no equivalentes o derivables del conjunto Λ original.

Estas perspectivas de conocimiento así calculadas podrían ser usadas de diversas formas para apoyar las actividades de investigación científica en el marco de las diversas corrientes epistemológicas existentes, por ejemplo, para corroborar una teoría usando el método inductivo en el marco del positivismo lógico [3], para falsar una teoría en el marco del racionalismo deductivo [23], o para construir perspectivas múltiples sobre las mismas fuentes de datos en el marco de diversas corrientes que defienden la pluralidad de visiones en ciencia [20], [11],[21].

3 Implementación de Perspectivas de Conocimientos en Grids de Datos

Las *Perspectivas de Conocimiento* pueden ser representaciones de teorías o conjeturas que realizan investigadores o grupos de investigación sobre conjuntos de datos. En este sentido, es importante que exista un método flexible para que los usuarios puedan definirlas.

El Grid es el escenario ideal para este tipo de procesamiento de datos porque al ofrecer altos niveles de seguridad y virtualización, permite que los usuarios puedan instalar nuevos servicios de datos, localizados en las mismas fuentes de datos. Estos nuevos servicios son la expresión de la perspectiva de conocimiento sobre los datos originales. El procesamiento de los datos en el lugar donde están alojados ofrece varias ventajas:

- facilita el procesamiento de grandes volúmenes de datos, evitando los retardos que produce la transferencia a otros sitios para su procesamiento
- permite el procesamiento de datos en lugares donde hay restricciones legales para su transferencia
- facilita la utilización de la perspectiva de conocimiento por parte de otros usuarios, o por la comunidad científica en general

- los nuevos servicios de datos podrían actualizarse más fácilmente a medida que la fuente de datos evoluciona

En este sentido, un proveedor de datos no solamente ofrecería un servicio plano de acceso a los datos, sino que también ofrecería un mecanismo para procesarlos *in-situ* y un servicio de alojamiento de nuevos servicios, con sus estructuras de datos, instalados por usuarios autorizados.

3.1 Arquitectura

La arquitectura que se propone permite instalar nuevos servicios de acceso y consulta a los datos. Estos nuevos servicios se soportan en el procesamiento de los datos originales, que denominamos datos crudos, y permiten crear una perspectiva adicional de los mismos. En la figura 1 se muestran los niveles de elaboración producto de procesamientos sucesivos sobre los datos que nuestra arquitectura permite a los usuarios.

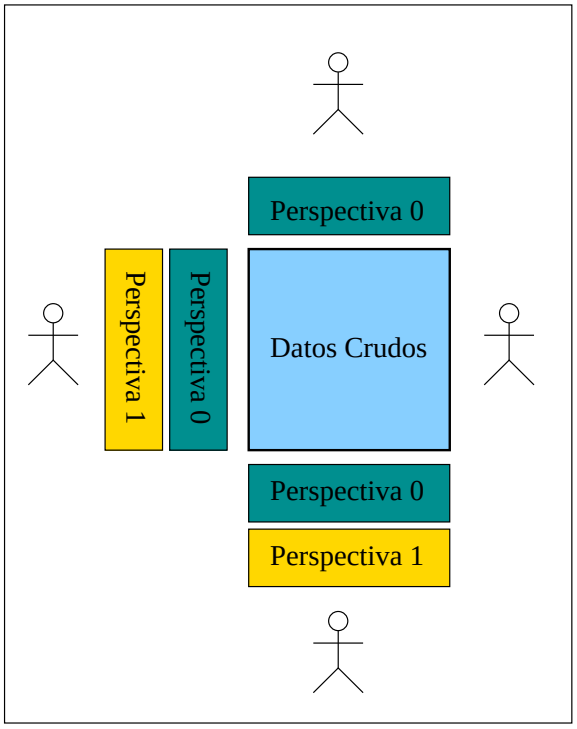


Figure 1: Niveles de Perspectivas

Los datos crudos son comunes a las diversas perspectivas que puedan ser instaladas sobre ellos, y no son afectados por éstas. El nivel de *perspectiva 0* corresponde al proceso de marcado ontológico y supone el desarrollo de métodos que definen una correspondencia entre los datos y los predicados básicos definidos por el usuario a través de la ontología. Estos métodos permiten construir el conjunto Ω. Ω representa un conjunto de índices sobre los datos, de tal modo que para cada predicado (propiedad o relación) podemos identificar que objetos representados en la fuente de datos lo satisfacen. El nivel de *perspectiva 1* es el resultado del cálculo del conjunto Ω' a partir de las conjeturas contenidas en Λ. Ω' representa un segundo nivel de elaboración, que permite crear índices adicionales con base a las conjeturas contenidas en Λ, para las propiedades que se deducen a través de inferencias lógicas. Es importante notar que todos los usuarios comparten los mismos datos crudos, pero que los niveles de perspectivas 0 y 1 son completamente definidos por ellos.

En la figura 2 se muestra la arquitectura desde el punto de vista de los servicios que deben ser implementados en el Grid para poder ofrecer el servicio de perspectivas. Hemos definido las interfaces requeridas en cada nivel y la semántica de operación que les corresponde. De esta forma cada ambiente Grid puede hacer diversas implementaciones del mismo servicio de un modo que en principio permitirá la virtualización del servicio. Las interfaces definidas permiten el uso de este concepto en forma distribuida, con lo cual el

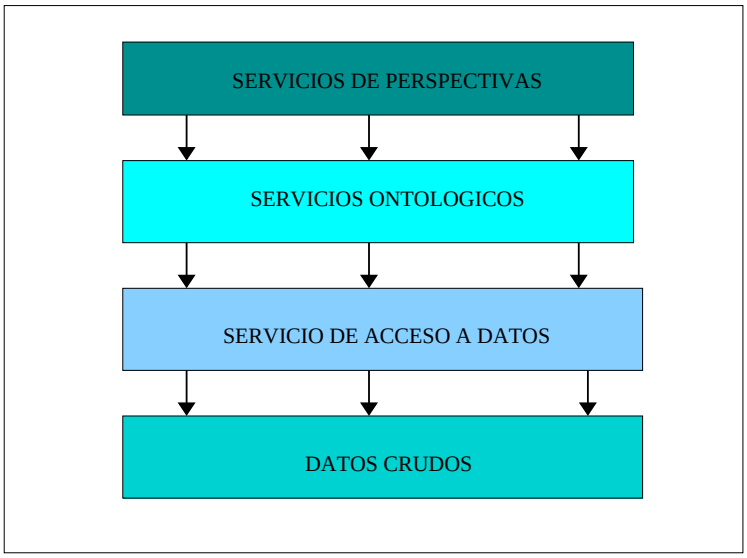


Figure 2: Niveles de Servicios

usuario puede definir e instalar perspectivas distribuidas. La arquitectura propuesta consta de tres capas de servicios,

- La capa de *servicios de acceso a datos* define las interfaces básicas y los servicios requeridos para acceder a los datos crudos (getData, putData, etc). Esta capa de servicios es generalmente instalada por el dueño de los datos y permite abstraer las primitivas básicas del formato en que se encuentra la fuente de datos. Por ejemplo, los datos podrían estar como una base de datos relacional o como un repositorio de objetos XML. La arquitectura que se propone permite que los mismos datos puedan ser accedidos a través de implementaciones diversas que pueden haber sido optimizadas para una fuente de datos en particular.

- La capa de *servicios ontológicos* define los servicios y las interfaces requeridos para la creación, almacenamiento y manipulación de ontologías. (p.e. UploadOntology, CheckOntology, etc). Usando los servicios de este nivel, el usuario primero describe una ontología adecuada para su perspectiva, es decir, hace una descripción de los objetos, sus propiedades y relaciones. Posteriormente desarrolla los métodos necesarios para aplicar estos predicados a los datos y construir el primer nivel de perspectiva (*perspectiva 0*). Finalmente desarrolla las aseveraciones (conjunto de conjeturas o inferencias lógicas) que aplicará a los datos para los cuales ya existe el nivel de *perspectiva 0* para construir el nivel de *perspectiva 1*.

- Finalmente la capa de *servicios de perspectivas* define las interfaces básicas requeridas para la creación, almacenamiento y manipulación de perspectivas de ambos niveles. (p.e.MakePerspective, QueryPerspective). La ejecución de estos servicios materializa los dos niveles de índices a partir de los datos crudos y la ontología que provee el usuario. El primer nivel de procesamiento establece una correspondencia entre los objetos contenidos en la fuente de datos y los predicados que satisfacen. El segundo, usa las aseveraciones lógicas contenidas en Λ para inferir nuevos predicados aplicables a las fuentes de datos.

Es importante notar que las implementaciones de los servicios de perspectivas tienen en común esas interfaces y las operaciones que representan, pero pueden corresponder a implementaciones diversas.

En la tabla 1 se mencionan algunos de los métodos implementados en las capas de servicios ontológicos y de perspectivas. *UploadOntology* permite al usuario cargar una especificación de una ontología (*OntologySpec*) hecha en un lenguaje ontológico, por ejemplo en RDF. *CheckOntology* permite al usuario hacer una verificación de una ontología previamente cargada. Los servicios *GetProperties* y *GetObjectsByPredicate* están principalmente orientados a ser ejecutados por los servicios de perspectivas. *GetProperties* devuelve

Servicios Ontológicos	Servicios de Perspectivas
UploadOntology(OntologyName,OntologySpec) CheckOntology(OntologyName) GetProperties(Class) GetClassesByPredicate(Predicate)	MakePerspective0(P0_Name, DataRef, P0_Spec) MakePerspective1(P1_Name,P0_Name,OntologyName) QueryPerspective(P_Name,Query)

Table 1: Interfaces y Métodos (lista no exhaustiva)

las propiedades que aplican a una determinada clase de objeto definido por la ontología. *GetClassesByPredicate* devuelve todas las clases a las que es aplicable un determinado predicado. En el nivel de servicios de perspectivas, el servicio *MakePerspective0* permite hacer el marcado ontológico de los objetos contenidos en la fuente de datos referenciada por *DataRef*. Este servicio recibe también como parámetro una estructura de datos (*P0_Spec*) que contiene un objeto con todos los métodos que implementan el marcado ontológico y una lista de pares*(Method,Predicate)*, en el que se asocia un método a un predicado. Inevitablemente los métodos deben estar adaptados a las estructuras de datos utilizadas en la fuente de datos. El servicio *MakePerspective1* permite asociar una ontología al nivel 0 de perspectiva, materializando los índices que representan inferencias lógicas sobre los predicados básicos usados en el marcado ontológico. Finalmente, *QueryPerspective* permite hacer consultas a las perspectiva, usando un lenguaje de consulta, por ejemplo, SQL.

4 Una implementación del servicio de Perspectivas en SUMA/G

A continuación se muestra la arquitectura de implementación del concepto de *Perspectivas de Conocimientos* en SUMA/G [2] ,una plataforma grid que ejecuta bytecode de Java en máquinas remotas en forma transparente, la cual está basada en servicios grids (Servicios Globus) [28]. En esta plataforma se ha implementado el Metaservicio para Instalación de Servicios en Grids (SIMG). Un *servicio* en este contexto está constituido por un *nombre de servicio*, una lista de servicios de los cuales depende el nuevo servicio (*requerimientos*), un *API*, una *documentación* (en la implementación actual un archivo javadoc) y una *implementación*, i.e., un conjunto de paquetes que implementa el servicio. Con sólo ejecutar un comando, un usuario podría instalar remotamente un objeto Java, por ejemplo, cuyo método constructor recorrería los objetos originales del repositorio de datos para hacer un marcado ontológico de éstos. El marcado ontológico se podría realizar a través de la creación de unos índices. Los otros métodos del objeto podrían permitir el acceso a los datos usando los índices, es decir, implementarían las diferentes funciones Φ_i.

4.1 Arquitectura de SUMA-onto en el Agente de Ejecución

Se utilizó la capacidad de instalación de servicios ofrecida por SUMA/G(SIMG) para definir el servicio de *perspectivas* como un conjunto de interfases que permiten implementar el concepto de *perspectivas de conocimiento*.La arquitectura del servicio de *perspectivas* en el agente de ejecución es mostrada en la figura 3.

En la actualidad los servicio de perspectivas, datos y ontologías, han sido definidos e instalados a nivel del agente de ejecución de SUMA/G, aunque la arquitectura futura del sistema incluirá un *proxy* especializado que permitirá la consulta a fuentes de datos distribuidas que han sido procesadas con el servicio de perspectivas. La facilidad de procesar la perspectiva y la consulta con el operador *submit* de SUMA/G permite desarrollar consultas *fuera de línea* cuyos resultados se almacenan temporalmente en los agentes de ejecución y que pueden ser solicitados por el usuario a través del mediador en cualquier momento.

Para probar el concepto de *perspectivas de conocimiento* se ha desarrollado una implementación de las interfases antes mencionadas. En esta implementación en particular la fuente de datos es una base de datos Mysql y las interfases del servicio de datos son implementadas usando JDBC y SQL. Para la implementación de los servicios ontológicos se usó JENA, un paquete que incorpora una serie de facilidades para manipular ontologías descritas usando la sintáxis de RDF (Resource Description Framework) y una serie de motores de inferencia aplicables a diversos lenguajes ontológicos (OWL, DAML-OIL, etc).

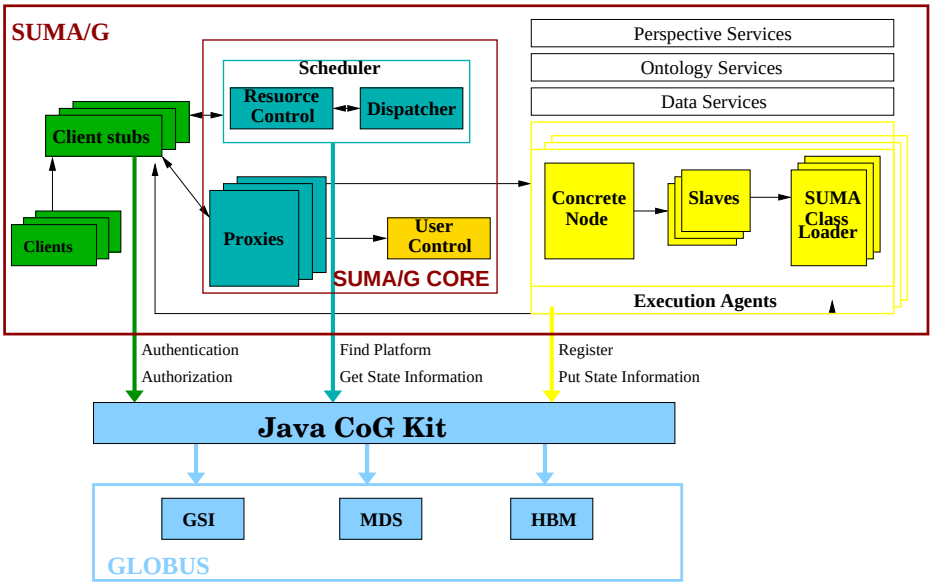


Figure 3: Arquitectura de SUMA-onto en el Agente de Ejecución

5 Un ejemplo usando Wordnet

Como prueba de concepto hemos hecho un ejemplo orientado a mejorar la recuperación de artículos científicos contenidos en una base de datos Mysql. Hemos usado Wordnet, un sistema en línea de referencia lexicográfica [9]. La base de datos fue instalada en un agente de ejecución de SUMA/G, en el cual se instalaron los servicios de acceso a datos, ontologías y perspectivas descritos en la sección 3.1. Usamos una versión de Wordnet hecha en Prolog y diseñamos un metaintérprete que recibe como entrada una palabra en inglés y produce recursivamente como salida un archivo RDF que representa el subárbol de palabras que son sus hipónimas. Esta ontología producida automáticamente con el metaintérprete puede ser manipulada directamente usando las primitivas y métodos de los servicios de perspectivas y ontologías respectivamente. Usando el servicio de perspectivas producimos un índice que apunta a los artículos que mencionan en su título cualquiera de las palabras contenidas en el árbol.

El metaintérprete usa los operadores s de *sentido* y hyp de *hiponimia* para producir automáticamente la ontología que usamos al construir la perspectiva. La ontología es producida en un formato manejable por los servicios ontológicos que hemos desarrollado.

La ontología contiene un conjunto de aseveraciones del tipo *Todo P es Q*, expresado en lógica proposicional a través de la relación de subclase entre P y Q o su equivalente en lógica de predicados:

$$\forall X \, P(X) \rightarrow Q(X)$$

para cada una de las relaciones de hiponimia.

Con el objeto de hacer un procesamiento de la fuente de datos que sea consistente con la ontología que se usa en el marco de la construcción de una perspectiva es necesario indicar cuales son los objetos de interés (para este ejemplo son títulos de artículos científicos) y para estos, que *significa* que un objeto X_i cualquiera *satisfaga* el predicado $P(X)$. Es decir, como podemos reconocer los diversos objetos y sus propiedades. En la práctica es necesario que definamos para cada fuente de datos a la que deseamos aplicar una perspectiva:

- Cuáles son los objetos allí representados, es decir, las cosas que hay, y cómo reconocerlas en un formato particular.
- Cuáles son las propiedades básicas de esos objetos, es decir, cómo podemos reconocer en un objeto una determinada propiedad básica.

En nuestro caso de ejemplo la fuente de datos es una base de datos relacional que contiene artículos científicos. Las tablas principales de esta fuente de datos y sus relaciones se muestran en la figura 4.

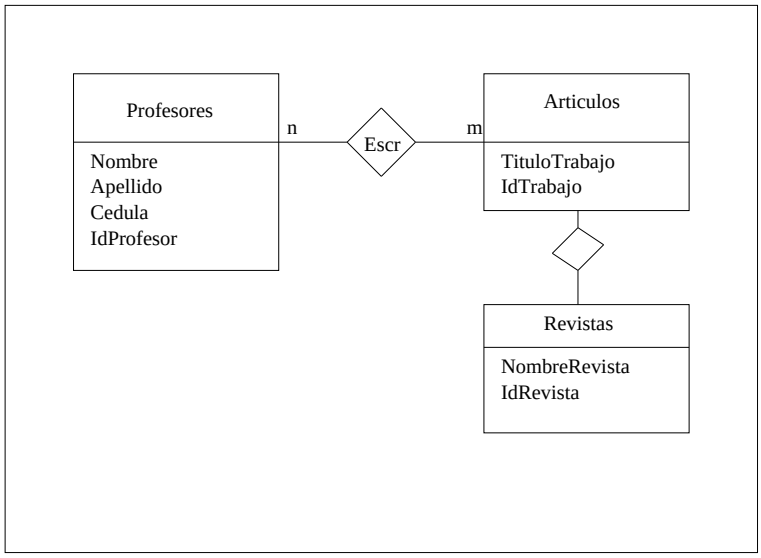


Figure 4: Arquitectura de la Base de Datos

Cada tabla en esta base de datos representa un tipo de objeto del dominio de discurso. Usamos una ontología (llamada *science* [13]) para describir los objetos representados en la fuente de datos y desarrollamos un esquema RDF para describir en forma genérica cualquier base de datos relacional, de tal forma que la descripción de las operaciones requeridas para desarrollar la perspectiva son desarrolladas con base a la ontología. Esta técnica es usada para clarificar la semántica que se asigna en forma implícita a los nombres de tablas, de columnas y de relaciones contenidas en la fuente de datos. En la figura 5 se muestra un fragmento de la descripción hecha de la base de datos.

```
<rdf_:Tabla rdf:about="&rdf_;kb_db_00055"
 rdf_:Nombre_de_tabla="Profesores"
 rdf_:Representa="Science:Academic−Staff"
 rdf_:tamano="7912"
 rdfs:label="kb_db_00055">
<rdf_:tiene_atributos rdf:resource="&rdf_;kb_db_00056"/>
<rdf_:tiene_atributos rdf:resource="&rdf_;kb_db_00058"/>
<rdf_:tiene_atributos rdf:resource="&rdf_;kb_db_00059"/>
<rdf_:tiene_atributos rdf:resource="&rdf_;kb_db_00060"/>
</rdf_:Tabla>
<rdf_:Atributo rdf:about="&rdf_;kb_db_00056"
 rdf_:Longitud_atributo="50"
 rdf_:Nombre_Atributo="Science:First−Name"
```

Figure 5: Descripción RDF de la Base de Datos

Con esta descripción indicamos qué clase de objetos contiene nuestra base de datos y que representan (*significan*) usando para ello una ontología de referencia (*science*). La propiedad que nos interesa identificar en el marco de este ejemplo son los artículos que hablan de un determinado tema, donde el *tema* es tomado de un vocabulario limitado (el subárbol de relaciones de hiponimia construido automáticamente con Wordnet). En el marco de este ejemplo, decimos que un artículo tiene la propiedad deseada si su título contiene la palabra. Usando esta definición construimos un índice que, para cada palabra del subárbol contenida en un artículo inserta una tupla en la tabla índice. Luego es completada para las relaciones de hiponimia contenidas

en Wordnet y generadas automáticamente en formato RDF.

En esta implementación, cuando procesamos una perspectiva, construimos un índice sobre los datos, con el objeto de interpretarlos en el marco de una determinada teoría. Cada aseveración de la teoría que usamos para procesar la perspectiva (cada aseveración A_i contenida en Λ) genera una tabla, con un número de columnas directamente relacionado con el número de objetos y propiedades mencionados en ella. Todas las tablas n-arias compatibles (que aceptan los mismos objetos en las mismas posiciones de diversas propiedades) son unidas en una sola con objeto de hacer mas eficiente el almacenamiento y posterior procesamiento de la perspectiva. En nuestro ejemplo, cada relación de hiponimia corresponde a una aseveración de la teoría. De esta forma, las siguientes aseveraciones contenidas en el subárbol de la palabra *biology*:

$$\forall(X) Embriology(X) \rightarrow Biology(X)$$

$$\forall(X) Botany(X) \rightarrow Biology(X)$$

$$\forall(X) Phytology(X) \rightarrow Biology(X)$$

Producen una tabla como la mostrada en la figura 6:

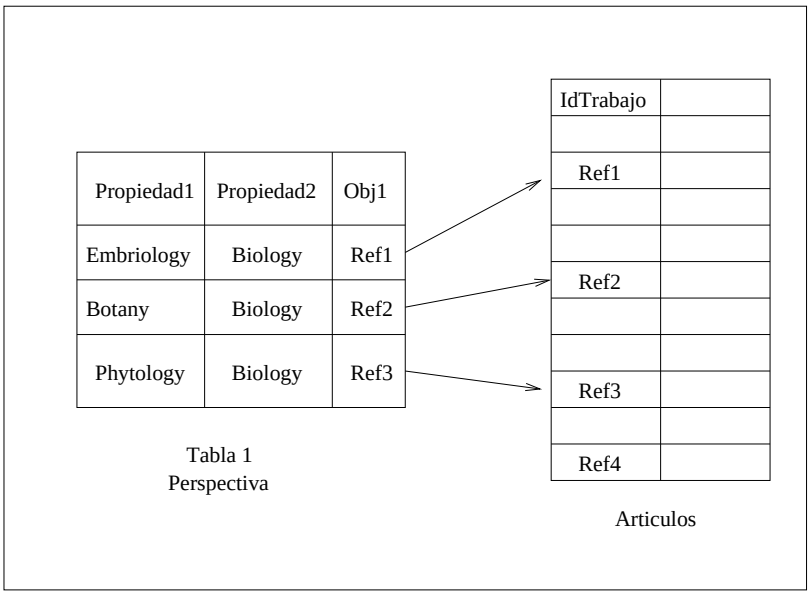


Figure 6: Perspectiva

Para este ejemplo nuestra perspectiva $\Pi = (\Gamma, \Omega, \Lambda)$ se define de la siguiente forma:

- Γ contiene artículos científicos. Para identificar las propiedades y objetos relevantes que la tabla representa hemos usado una descripción basada en un esquema RDF creado para este fin y la ontología *Science*.
- Ω son todos los artículos X_i que satisfacen el predicado *Biology*. $(Biology(X_i))$.
- Ω' corresponde a todos los artículos X_i que agregamos por satisfacer las aseveraciones en Λ.
- Λ contiene las relaciones de hiponimia directas e indirectas del predicado *Biology*.

Se construye así un índice por cada término de la ontología descrita por *science*, esto corresponde al marcado ontológico del nivel de *perspectiva 0*. Luego se usan las relaciones de hiponímia para agregar relaciones entre los palabras contenidas en los índices lo que corresponde al nivel de *perspectiva 1*. .Un ejemplo más elaborado, por ejemplo con imágenes, necesitará un marcado ontológico distinto, que relacione las imágenes con los predicados.

6 Trabajos Relacionados

El concepto de perspectivas de conocimiento aqui planteado construye un puente entre técnicas que son usadas en el ámbito del web semántico [27] y técnicas planteadas para el procesamiento de datos en Grids de Datos [5] [26]. Recientemente ha surgido un gran interés por parte de la comunidad que desarrolla las tecnologías del Grid en los desarrollos del Web Semántico, constituyendo una corriente que ha sido bautizada como el Grid Semántico [14][1] [4][6][22]. Algunos de estos esfuerzos se han orientado a una solución basada en el uso de agentes computacionales [25]. El Web Semántico trabaja en desarrollar técnicas para, a través del uso de ontologías, mejorar el tipo y calidad del conocimiento que puede obtenerse a través del web [10].

Para ello se han desarrollado familias de lenguajes y técnicas de razonamiento que permiten describir recursos, servicios y razonar sobre ellos [8][19] [18][7]. La capacidad de describir estos recursos y servicios en formas mas elaboradas permiten combinarlos para el desarrollo de nuevos servicios o para la obtención de mas información.

Comparativamente, el Grid ofrece ventajas sobre el web que están relacionadas con la seguridad y con la capacidad de integrar servicios y datos alrededor de organizaciones virtuales. Los esfuerzos del grid semántico para agregar conocimiento al grid a través de técnicas que tienen su origen en el campo de la inteligencia artificial han sido clasificados en dos grandes grupos, conocimiento sobre el grid y conocimiento para el grid [15]. El primero de ellos busca describir recursos y servicios en formas sofisticadas para hacerlo interoperables, el segundo busca extraer mas conocimientos a través del uso de recursos ontológicos para describir y descubrir nuevas relaciones entre los datos.

Estos esfuerzos requieren de servicios de acceso a datos y de creación de metadatos que están siendo desarrollados. Parte de estos esfuerzos aprovechan la experiencia acumulada en el ámbito de las bibliotecas digitales. En esa comunidad es común el uso de metadatos y de etiquetas semánticas para organizar la información [12].

El uso de ontologías para mejorar la recuperación de datos e información ha sido ampliamente estudiado. En particular Wordnet ha sido usado para reducir la ambiguedad en las búsquedas a través del uso de las relaciones semánticas entre palabras [29].

Nuestra propuesta tiene algunas características que la diferencian de los esfuerzos antes mencionados:

- Desarrolla un concepto que vincula teorías lógicas descritas a través del uso de ontologías con un modelo de procesamiento de datos. Esta vinculación expone claramente la relación entre las teorías y el modo de procesar y organizar los datos. En un primer nivel (*perspectiva 0*) facilita el marcado semántico de cualquier fuente de datos. En un segundo nivel (*perspectiva 1*) permite inferir nueva información en forma consistente con la teoría de un modo que permite verificar las conjeturas que fueron aplicadas a los datos para procesarlos. En ambos casos la fuente de datos original queda inalterada.

- El marcado y procesamiento ontológico es definido completamente por el usuario en un ambiente grid. De esta forma permitimos que cada usuario procese los datos a su conveniencia.

- El procesamiento se hace donde están alojados los datos, evitando los costos o dificultades propias asociadas a transferir grandes archivos para poder procesarlos.

- Desde el punto de vista de la descripción de conocimiento sobre el grid, permite describir recursos, servicios y fuentes de datos al igual que integrarlas en una forma que puede ser completamente definida por cada usuario y no depende del dueño de los datos o del proveedor del servicio.

- Propone un conjunto nuevo de servicios para manipulación de ontologías.

- Describe los servicios requeridos en el grid para implementar operativamente el concepto de *perspectiva*.

Hemos puesto el énfasis en que el usuario tenga el control en todo momento de su perspectiva de los datos, aunque éstos no le pertenezcan. Creemos que de esta forma se puede facilitar la diversidad de visiones sobre hechos científicos.

7 Conclusiones y Trabajo Futuro

En este trabajo se presenta una metodología que establece un puente entre las técnicas de manipulación de datos basadas en criterios ontológicos y las técnicas de manipulación virtualizada y segura de grandes

volúmenes de datos que ofrecen los Grids de Datos. Se propone el concepto de *Perspectivas de Conocimiento* como una base conceptual para manipular datos e información experimental en este contexto. Esta interpretación conduce a la creación de índices, por parte de cada usuario, que le permiten acceder a la data en forma consistente con la teoría usada para indexarla.

Los resultados iniciales del uso de esta metodología, reportados en este trabajo, muestran que en escenarios donde la complejidad de la información no es elevada, este mecanismo permite la generación de índices *in situ* por parte de los usuarios. Esto significa que la metodología propuesta es potencialmente útil, al menos, para el establecimiento de conjeturas científicas relativamente sencillas, que sean parte de teorías o sistemas de conjeturas mucho más complejos. Desde el punto de vista de representación de conocimientos, proponemos la definición de ontologías, que describen las operaciones requeridas para desarrollar la perspectiva en un entorno distribuido y evolutivo. El entorno de Grid permite que el usuario tenga la autorización y suficientes recursos computacionales para la generación de índices de datos y servicios en el lado del servidor que contiene el repositorio de datos. La combinación de ambos aportes permite la definición de servicios que implementan nuevas interfases de acceso a los datos, de acuerdo a una determinada perspectiva de agrupación o clasificación de éstos, lo que facilita la generación de nuevo conocimiento.

El uso de ontologías para la descripción de la fuente de datos distribuida facilita la mediación de los datos y el acceso uniforme a fuentes de datos heterogéneas, lo que se tiene previsto abordar en un futuro inmediato. Otra línea de trabajo inmediato es que aplicaremos esta metodología en escenarios de mayor complejidad, por ejemplo, en escenarios donde los objetos son imágenes y los predicados asociados a éstas se satisfacen por medio de algoritmos de procesamiento de imágenes.

References

[1] Mario Cannataro and Domenico Talia. Semantics and knowledge grids: Building the next-generation grid. *IEEE Intelligent Systems*, 19(1):56–63, 2004.

[2] Y. Cardinale and E. Hernández. Parallel Checkpointing on a Grid-enabled Java Platform. *Lecture Notes in Computer Science*, (European Grid Conference EGC2005), February 2005. To appear.

[3] Rudold Carnap and Laura De Shrenk. *La construcción lógica del Mundo.* Universidad Nacional Autónoma de México. ISBN 968360529X, 1988.

[4] L. Chen, N.R. Shadbolt, F. Tao, C. Puleston, C. Goble, and S.J. Cox. Exploiting semantics for e-science on the semantic grid. In *Web Intelligence (WI2003) workshop on Knowledge Grid and Grid Intelligence*, pages 122–132, 2003.

[5] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, (23):187–200, 2001.

[6] D. De Roure and J.A. Hendler. E-science: the grid and the semantic web. *IEEE Intelligent Systems*, 19(1):65–71, 2004.

[7] M. Dean et al. Owl: Web ontology language 1.0 reference. Technical report, World Wide Web Consortium, 2002.

[8] S. Decker et al. The semantic web: The roles of xml and rdf. *IEEE Internet Computing*, 15(3):63–74, October 2000.

[9] Christiane Fellbaum. *Wordnet: An Electronic Lexical Database.* MIT Press, 1999.

[10] Dieter Fensel, J. Handler, H. Lieberman, and W. Wahlster. *Spinning the semantic Web : Bringing the World Wide Web to its full potential.* Cambridge, Mass. : MIT Press., 2005.

[11] Paul Feyerabend. *"Problems of Empiricism", Beyond the Edge of Certainty: Essays in Contemporary Science and Philosophy.* Prentice Hall, 1965.

[12] G. Fox. Data and metadata on the semantic grid. *Computing in Science & Engineering*, 5(5):76–78, 2003.

[13] F Freitas. Ontology of science. Technical report, Universidade Federal de Santa Catarina, 2001.

[14] C. Goble and D. De Roure. The semantic web and grid computing. In V. Kashyap and L. Shklar, editors, *Real World Semantic Web Applications*, volume 92 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2002.

[15] C.A. Goble, D. De Roure, N.R. Shadbolt, and A.A.A. Fernandes. Enhancing services and applications with knowledge and semantics. In I. Foster and C Kesselman, editors, *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann, 2004.

[16] Emilio Hernández, Yudith Cardinale, Carlos Figueira, and Alejandro Teruel. SUMA: A Scientific Metacomputer. In *Parallel Computing: Fundamentals and Applications. Proceedings of the International Conference ParCo99*, pages 566–573. Imperial College Press, 2000. ISBN 1-86094-235-0.

[17] T. Hey and A Trefethen. "e-science and its implications". *Philosophical Transactions of the Royal Society*, 361(1809):1809–1825, 2003.

[18] I. Horrocks. Daml-oil: A reason-able web ontology language. In *Proceedings of EDBT*, number 2287 in Lecture Notes in Computer Science, pages 2–13. Springer, March 2002.

[19] Ian Horrocks, Dieter Fensel, Jeen Broekstra, Stefan Decker, Michael Erdmann, Carole Goble, Frank van Harmelen, Michel Klein, Steffen Staab, Rudi Studer, and Enrico Motta. OIL: The Ontology Inference Layer. Technical Report IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences, September 2000. See http://www.ontoknowledge.org/oil/.

[20] Thoma Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, 1962.

[21] Imre Lakatos. *The Methodology of Scientific Research Programmes: Philosophical Papers Volume 1*. Cambridge University Press, 1970.

[22] S. Newhouse, S. Mayer, S. Furmento, S. McGough, J. Stanton, and J. Darlington. Laying the foundations for the semantic grid. In *AISB Workshop on AI and Grid Computing.*, 2002.

[23] Karl Popper. What can logic do for philosophy? *Aristotelian Society*, 22:141–154, 1948.

[24] Karl R. Popper. *The Logic of Scientific Discovery*. Routledge, 1962.

[25] Omer F. Rana and Line Pouchard. Agent based semantic grids: Research issues and challenges. *Journal of Parallel and Distributed Computing Practices*, 2003.

[26] Ben Segal. Grid computing: The european data grid project. *IEEE Nuclear Science Symposium and Medical Imaging Conference*, 2000.

[27] O. Lassila T. Berners-Lee, J. Hendler. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[28] The Globus Alliance. The Globus Toolkit. http://www.globus.org/.

[29] Ellen M. Voorhees. Using wordnet to disambiguate word senses for text retrieval. In Robert Korfhage, Edie M. Rasmussen, and Peter Willett, editors, *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1, 1993*, pages 171–180. ACM, 1993.

Verificación de Restricciones de Integridad en Transacciones Distribuidas sobre un Cluster de Bases de Datos Relacionales*

Stephane Gançarski[1], Claudia León[2], Hubert Naacke[1], Marta Rukoz[2] y Pablo Santini[2]

[1]Laboratoire d'Informatique Paris 6, Université Pierre et Marie Curie.
8 Rue du Capitan Scott, 75015, Paris, France.
{Stephane.Gancarski,Hubert.Naacke}@lip6.fr

[2]Centro de Computación Paralela y Distribuida, Universidad Central de Venezuela.
Apdo. 47002, Los Chaguaramos, 1041 A, Caracas, Venezuela.
Telefóno: (58) 212 6051287 Fax: (58) 212 6051131
{cleon,mrukoz}@ciens.ucv.ve, psantini@eda.com

Resumen

En este trabajo se proponen mecanismos para mantener restricciones de integridad referencial y restricciones globales conjuntivas en sistemas de multibases de datos relacionales y se presentan los resultados obtenidos en la evaluación de estos mecanismos en un ambiente experimental desarrollado sobre un cluster con bases de datos homogéneas Oracle*9i*.

La evaluación tuvo como objetivo medir el tiempo que consume realizar el chequeo de una restricción global conjuntiva en un sistema distribuido.

Los resultados experimentales muestran la factibilidad de aplicar los mecanismos propuestos para verificar restricciones globales en multibases de datos relacionales ya que el overhead introducido en el chequeo distribuido se reduce hasta en un 50% con respecto a un chequeo centralizado.

Palabras claves: Multibases de datos, Restricciones de Integridad, Sistemas de transacciones, Sistemas distribuidos.

Abstract

This paper presents a solution to check referential integrity constraints and conjunctive global constraints in a relational multi database system. It also presents the experimental results obtained by implementing this solution over a PC cluster with Oracle*9i* DBMS.

The goal of those experimentations is to measure the time spent to check global constraints in a distributed systems. The results shows that the overhead induced by our distributed constraint checking is reduced by 50% compared to a centralized checking of constraints.

Keywords : Multi Databases, Integrity Constraints, Transaction Systems, Distributed Systems.

Introducción

El establecimiento de la Internet, ha permitido que las funcionalidades y la complejidad de las aplicaciones informáticas, las cuales utilizan sistemas de bases de datos, haya crecido enormemente durante las últimas dos décadas. Como resultado de ello, el compartimiento de la información almacenada en bases de datos disímiles y la necesidad de proteger la calidad de la información son objetivos altamente demandados. Un sistema de multibase de datos (MDBS) es uno de los enfoques para proveer el compartimiento de información ó interoperabilidad entre múltiples sistemas de base de datos autónomos. La calidad de los datos es asegurada, por lo general, a través de la definición de restricciones de integridad, las cuales son aserciones lógicas que deben ser satisfechas por la BD.

En MDBS se requiere la definición de restricciones de integridad globales sobre los datos distribuidos. Dado que los datos están almacenados en diferentes sitios, es necesario determinar en dónde será almacenada y verificada cada restricción global con el objeto de minimizar la transferencia de datos. En [2] se establece una clasificación de restricciones de integridad y se propone para cada tipo de restricción identificado, estrategias para su verificación en MDBS OO que contemplan el tipo de actualización y los nodos sobre los cuales ocurren las actualizaciones.

En este trabajo proponemos mecanismos para verificar restricciones de integridad referencial y globales conjuntivas en MDBS relacionales. Igualmente mostramos la evaluación de la solución para restricciones globales conjuntivas en un MDBS de base de datos homogéneas Oracle*9i* sobre un Cluster de computadores.

El resto de este trabajo está organizado como sigue. La sección 1 presenta los conceptos fundamentales o contexto dentro del cual se propone este trabajo. La sección 2 describe los esquemas de solución que proponemos para verificar restricciones de integridad referencial y globales conjuntivas en MDBS relacionales. La sección 3 describe en detalle la experimentación realizada. La sección 4 presenta los resultados obtenidos. Finalmente se expone las conclusiones de este trabajo.

1. MultiBases de Datos, Transacciones Anidadas y Chequeo de Restricciones Globales

Un sistema de multibase de datos (MDBS) soporta operaciones sobre múltiples BD cada una gestionada por un manejador de base de datos. En estos sistemas, es posible tener diferentes arquitecturas y niveles de integración entre los componentes, correspondientes a diferentes niveles de servicios globales [8]. Una de las funcionalidades mas importantes a proveer en MDBS es el mantenimiento de restricciones de integridad globales, las cuales son restricciones definidas sobre objetos almacenados en diferentes bases de datos. En este trabajo proponemos mecanismos para mantener restricciones de integridad globales en un MDBS relacional homogéneo. Nuestra propuesta considera un sistema multibase con arquitectura clásica [8] que posee las características siguientes:

- Todas las transacciones son globales y manejadas por un manejador de transacciones globales (con la ayuda de un esquema global que describe cada uno de los datos y su localización) aún cuando se ejecuten completamente sobre un único sitio. Este manejador debe garantizar el control de concurrencia entre transacciones y entre sub-transacciones de una transacción.
- Se utiliza el modelo de transacciones anidadas propuesto por Moss [6]. Una Transacción Anidada puede contener un número aleatorio de sub-transacciones y cada sub-transacción a su vez puede estar compuesta de otro número aleatorio de sub-transacciones. Esto resulta en una jerarquía de profundidad arbitraria de transacciones anidadas, que puede ser representada por un árbol de transacciones. En su trabajo Moss [6] describe en detalle las propiedades del modelo de transacciones anidadas así como los algoritmos para control de concurrencia y validación. También presenta una simplificación donde sólo las transacciones hojas acceden a los objetos, cada hoja es ejecutada en un sólo sitio y sólo accede a datos en dicho sitio. Esto permite simplificar los procesos de control de transacciones sin restar expresividad al modelo. En este trabajo tomamos este modelo transacciones anidadas de Moss.
- La asignación de las sub-transacciones de una transacción anidada a los diferentes sitios es inducida por la distribución de los datos. Cuando una transacción global es iniciada sobre un sitio S_i sus sub-transacciones son iniciadas recursivamente en ese mismo sitio. Cuando se detecta una sub-transacción con la totalidad de sus hojas ejecutándose sobre un mismo sitio, pero distinto del inicial, esta sub-transacción es enviada a dicho sitio donde será ejecutado completamente su subárbol. La Figura 1 ilustra esta política.

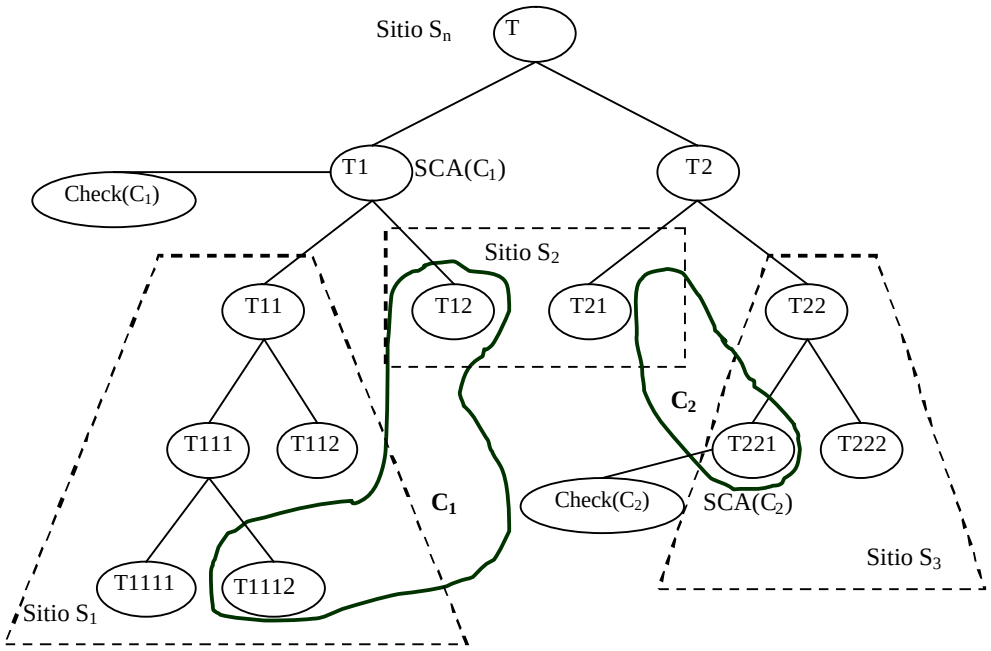


Figura 1. Distribución y restricciones tocadas por una transacción anidada.

2. Verificación de Restricciones de Integridad Globales en Transacciones Anidadas en Multibases de Datos Relacionales

En MBDS la tarea de mantener restricciones de integridad globales requiere determinar ¿Dónde será verificada cada restricción?. Esta decisión depende del tipo de restricción, del tipo de actualización y del sitio donde dicha actualización haya sido realizada. En [2] se proponen estrategias de verificación para ser aplicadas sobre sistemas de BD distribuidas OO que soportan transacciones anidadas, las cuales asumimos como punto de partida en este trabajo que tiene como objetivo aportar soluciones eficaces para el chequeo de restricciones de integridad globales en MDBS relacionales, en particular para restricciones de integridad referencial y globales conjuntivas. A continuación se describen las características más relevantes de las estrategias propuestas en [2] :

- El principio fundamental es que el control de la verificación de la restricción C_i es realizado por el ancestro común más joven de las hojas que tocan dicha restricción. Esta transacción es denotada $SCA(C_i)$.
- Dado que toda subtransacción mantiene el control de ejecución del subárbol del cual es raíz [6] es posible asegurar que en caso de violación de C_i no sólo serán abandonadas las transacciones que la tocan, sino también aquellas transacciones que puedan haber visto sus resultados. Por otra parte, y contrariamente a los sistemas que manipulan transacciones planas, las transacciones que no tengan relación alguna con C_i continúan su ejecución.
- Se tiene una comunicación directa entre las hojas que tocan una restricción C_i y la transacción $SCA(C_i)$, lo cual permite realizar la verificación lo más pronto posible, esto sucede inmediatamente después que todas las transacciones hojas que tocan C_i hayan terminado. En caso de una violación, la propagación de un mensaje de abandono hacia abajo en el correspondiente sub-árbol permite abandonar las sub-transacciones cuyos efectos deben ser deshechos para garantizar la integridad.
- La verificación de una restricción C_i se realiza sobre el conjunto de objetos modificados por la transacción anidada. Esto significa que el proceso de verificación de C_i, denotado Check(C_i), se ejecuta utilizando los efectos producidos por las transacciones hojas que tocan C_i y hayan decidido validar. Se denota $Efecto(T_j,C_i)$ a los objetos tocados por la transacción T_j e involucrados en la restricción C_i. El proceso de $Check(Ci)$ tendrá acceso privilegiado a $Efecto(T_j,C_i)$.
- Como consecuencia de la distribución de las sub-transacciones, el $SCA(C_i)$ está ubicado sobre el mismo sitio donde están todas las hojas que tocan C_i ó sobre el sitio donde se inició la transacción global (ver Figura 1).

- La ejecución de *Check(Ci)* es una transacción anidada hija de *SCA(*C_i*)*.

Para realizar la verificación de una restricción global en MDBS es necesario escoger el ó los sitios adecuados para ejecutar el proceso de verificación a fin de minimizar la cantidad y el tamaño de los mensajes a transmitir sobre la red. En estos casos es necesario considerar tanto la estructura de la restricción como los sitios sobre los cuales son ejecutadas las transacciones hojas que la tocan. Este problema es ilustrado en la Figura 1 en la cual están delimitados por líneas continuas gruesas los datos involucrados por cada restricción C_1 ó C_2 y las hojas que tocan esas restricciones. Así, C_1 involucra datos sobre los sitios S_1 y S_2 y es *tocada* por T_{1112} sobre S_1 y T_{12} sobre S_2. C_2 involucra datos sobre los sitos S_2 y S_3 y es *tocada* por T_{221} sobre S_3. Veamos a través de este ejemplo la necesidad de minimizar la transferencia de datos entre los sitios involucrados en el chequeo de una restricción global. Así :

- Para la restricción C_2, aunque la única hoja que la toca es ejecutada sobre S_3, es probable que el proceso de verificación sea ejecutado principalmente sobre S_2. Por ejemplo cuando C_2 sea una restricción referencial, su chequeo necesita comparar los datos modificados sobre S_3 por T_{221} contra los datos almacenados en S_2 asociados a la restricción. Evaluar C_2 en S_2 requiere sólo transferir los datos modificados en S_3 hacia S_2, mientras que su evaluación sobre S_3 puede requerir transferir todos los datos de S_2 involucrados en la restricción, incluso aquellos que no son afectados directa o indirectamente por la transacción.
- Para la restricción C_1 la situación es todavía más compleja. Su evaluación puede ser ejecutada en S_1, en S_2, o en ambos sitios, dependiendo de la naturaleza de la restricción y de las acciones realizadas por T_{1112} y T_{12}.

2.1 Uso de Conjuntos Intersitios

Generalmente, para optimizar el proceso de verificación de restricciones globales, se agrupan en el momento de la ejecución los objetos modificados por una transacción y se verifica la restricción sólo con respecto a esos objetos y aquellos relacionados a través de la restricción. En un contexto distribuido es necesario minimizar la transferencia de objetos sobre los cuales la restricción será verificada. Con este propósito en [2] se propone un análisis previo de cada *restricción global* para minimizar la cantidad de objetos a transmitir en conjuntos denominados *intersitios*. La generación de conjuntos *intersitios* comprende los dos pasos que se describen a continuación:

Primer paso: Está inspirado en el trabajo de Gupta y Widom [4] y consiste en determinar, para los sitios donde sea posible, un *predicado localmente evaluable* cuya satisfacción con respecto a los objetos modificados sobre ese sitio sea suficiente para garantizar la satisfacción de la restricción global. Ese predicado será designado en adelante por PL_i donde *i* denota el sitio S_i.

Segundo paso: Está basado en el método de Grufman y otros [3] quienes proponen una forma de descomponer una restricción global cuantificada universalmente sobre una conjunción de predicados a través de la definición de *predicados intersitios*. La idea es obtener una subrestricción para cada BD que describa la responsabilidad de dicha BD con respecto a la satisfacción de la restricción global, más una restricción particular que relaciona todos los *predicados intersitios* con la restricción a verificar. La última restricción es almacenada en un sólo sitio, en donde la verificación final de la restricción tendrá lugar. Existen tantas descomposiciones posibles para una restricción global universal como sitios estén implicados por ella. En la descomposición propuesta en [2] el conjunto asociado al predicado intersitio del sitio S_i contiene solamente los objetos modificados por una transacción que no satisfacen el predicado localmente evaluable asociado a S_i, esto con el objetivo de reducir la cardinalidad de dichos conjuntos.

2.2 Generación de Transacciones Anidadas para Realizar la Verificación

Para realizar la verificación de una *restricción global C*, el *SCA*(*C*) inicia en el sitio donde se encuentre (denotado $Sitio_{SCA(C)}$) el proceso *Check*(*C*). Este proceso será la raíz de una transacción anidada distribuida que tendrá una hija en cada sitio S_i donde existen hojas que toquen *C*. Cada hija, denotada *Check*(CS_i), está encargada de la verificación de *C* con respecto a las actualizaciones hechas en S_i. Cada subtransacción *Check*(CS_i) tiene el comportamiento siguiente:

1. Verifica el predicado localmente evaluable PL_i si existe. Si PL_i es satisfecho, no será necesario realizar otras acciones ya que se puede inferir que *C* se satisface con respecto a las modificaciones hechas en S_i.
2. Si PL_i no es satisfecho o no existe, construye el $intersitio_i$, es decir, selecciona los objetos sobre S_i necesarios para hacer la verificación de la restricción global en los otros sitios donde existan datos involucrados en *C*. En

este caso, envía a cada sitio S_j donde se debe verificar C, las modificaciones realizadas en S_i, contenidas en $intersitio_i$.

3. Por ultimo, inicia en S_j el proceso de verificación propiamente dicho que se denomina $Check(RG_{ji})$.

La transacción anidada *Check*(*C*) es distribuida sobre diferentes sitios: los sitios que contengan hojas que tocan *C* y los sitios donde se debe verificar *C*. La Figura 2 muestra la correspondiente transacción anidada cuando una restricción es tocada por sub-transasacciones en dos sitios, para el caso de la restricción C_1 de la Figura 1. Si la restricción es tocada por transacciones que se ejecutan en un único nodo (como la restricción C_2 en la Figura 1), la transacción anidada que se genera contiene sólo uno de los dos sub-árboles de la Figura 2. La sub-transacción $Check(PL_i)$ comprende la verificación del predicado local si lo hubiese y la generación del conjunto *intersitio*. La sub-transacción $Check(RG_{ji})$ corresponde a la verificación de la restricción global, con los datos modificados en el sitio i ($intersitio_i$) sobre la BD almacenada en el sitio j.

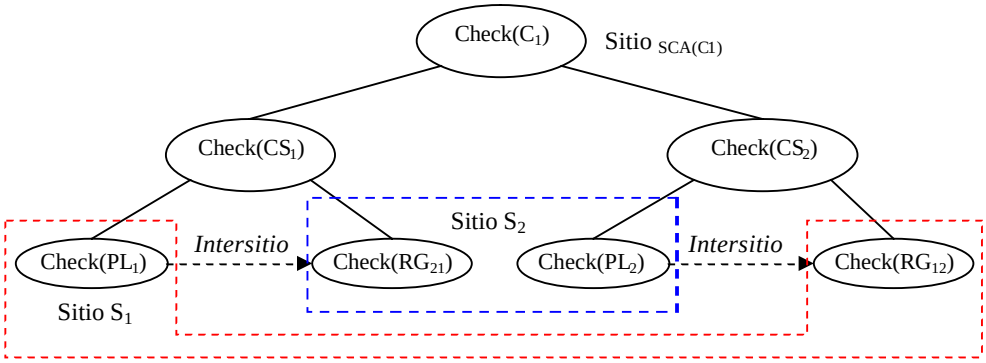


Figura 2. Verificación de restricción global tocada por transacciones sobre dos sitios.

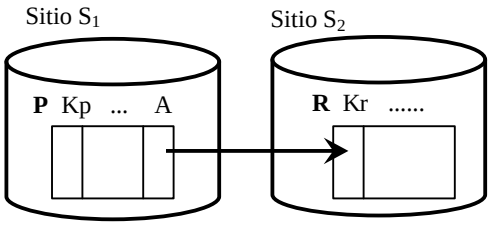
Una restricción global puede contener predicados y cuantificadores que necesitan una verificación simultánea sobre diferentes sitios, a través de transacciones que se intercambian información. Tales restricciones pueden tener una complejidad arbitraria, ya que pueden ser definidas a partir de cualquier tipo de predicado, de cuantificador y de conector, lo cual hace muy compleja la tarea de establecer un algoritmo general para la generación de la transacción anidada encargada de realizar el chequeo. Por esta razón, en este trabajo se restringe el estudio a dos clases de restricciones globales: las bien conocidas restricciones de integridad referencial [1] y otra clase de restricciones más general, que llamamos restricciones globales conjuntivas, las cuales aparecen cuando existen cadenas de relaciones entre datos, no necesariamente claves primarias, en diferentes bases de datos.

2.3.- Restricciones de integridad referencial

En MDBS los datos son distribuidos en diferentes BD donde los objetos son representados e identificados de forma distinta. Las restricciones de integridad referencial son definidas sobre la relación entre la clave primaria de una tabla y el atributo definido como *foreign key* en otra tabla. En sistemas centralizados los manejadores de BD se ocupan del mantenimiento de este tipo de restricciones, no siendo así en Multibases de datos. Este tipo de restricciones permiten identificar y relacionar objetos en diferentes BD y son un mecanismo idóneo para asegurar que las relaciones entre tablas de diferentes BD son coherentes.

```
Create Table P (
  Primary Key Kp,

  Foreign Key A
  Reference R(Kr),
);
```




```
Create Table R (
  Primary Key Kr,

);
```

Figura 3: Restricción de integridad referencial entre las tablas P y R

Considerando las bases de datos de la Figura 3 una restricción de integridad referencial C_f puede ser definida de la siguiente manera: $\forall p \in P, \exists r \in R,\ r.\mathrm{Kr} = p.\mathrm{A}$.

Obsérvese que una clave primaria en R no necesariamente es referenciada por la "*foreign key*" en P, sin embargo para toda "*foreign key*" en P debe existir el valor correspondiente como clave primaria en R. Para mantener la restricción C_f, se debe verificar que se satisface cuando se realiza un *Delete* o un *Update* en la tabla R, asi como también cuando se realice un *Insert* o un *Update* en la tabla P. A continuación se describe la generación de la transacción anidada que se encarga de verificar la restricción cuando se realice un *Delete* en R y un *Insert* en P.

2.31. Eliminación de clave primaria

Una restricción referencial puede ser definida bajo dos modalidades: *Restrict* o *Cascade*. Con la opción *Restrict* si el *Delete* en R (o *Update*) viola la integridad referencial, no se debe permitir la operación. Bajo la opción *Cascade* cuando se realiza un *Delete* (ó *Update*) en R, se deben eliminar (modificar) también todas las tuplas en la tabla P cuyo atributo "*foreign key*" referencia la clave primaria eliminada (modificada) en R.

La Figura 4 presenta el código pseudo-SQL y la correspondiente transacción que realiza la verificación para garantizar la restricción C_f cuando se realiza una operación: DELETE R WHERE <predicado> bajo la opción *Restrict* y la Figura 5 presenta el código pseudo-SQL y la correspondiente transacción para garantizar la restricción C_f bajo la opción *Cascade*.

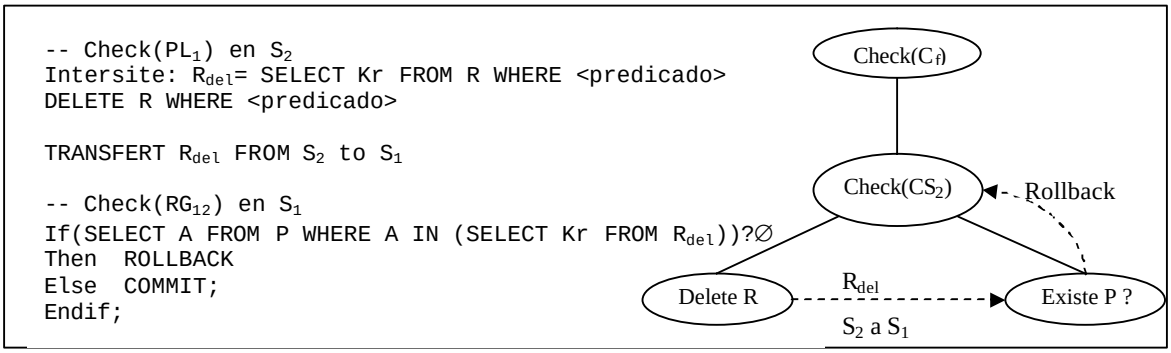


Figura 4: Verificación de una restricción referencial con opción *RESTRICT* tocada por una transacción que elimina claves primarias en tabla R

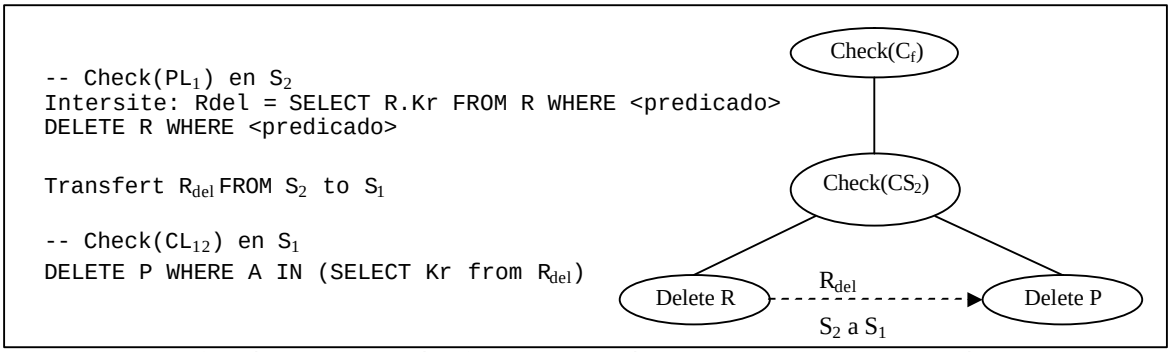


Figura 5: Verificación de una restricción referencial con opción *Cascada* tocada por una transacción que elimina claves primarias en tabla R

2.3.2. Inserción de una clave "foreign"

Cuando se inserta una entidad (o se modifica A) en la tabla P se debe asegurar la existencia de la clave primaria correspondientes en la tabla R. La Figura 6 presenta el código pseudo-SQL y la correspondiente transacción que garantiza la restricción C_f cuando se realiza una operación: INSERT INTO P (A) Values (<predicado>).

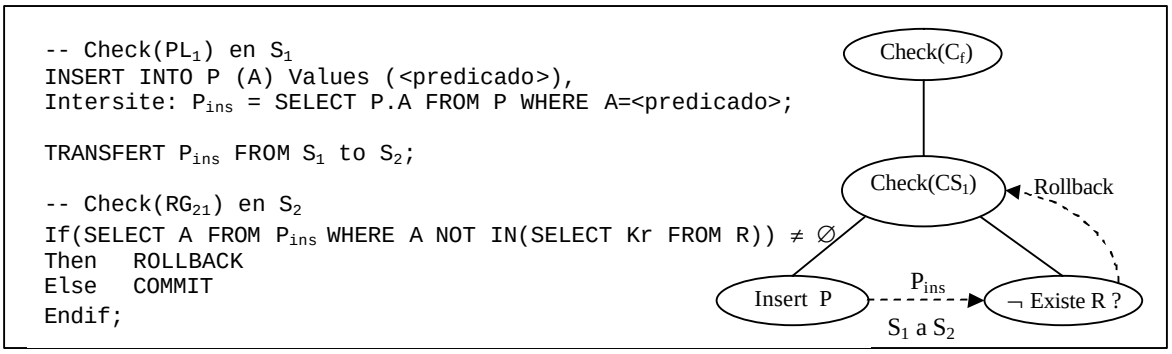


Figura 6: Verificación de una restricción referencial tocada por una transacción que inserta entidades en tabla P

En la verificación de restricciones de integridad referencial es necesario resaltar los siguientes aspectos:

- Mas que chequear la restricción, en el caso de un *Delete* en la tabla R bajo la opción Cascade, garantizamos que la restricción se satisfaga a través de la eliminación de todas las entidades en P con la misma clave.
- No es posible, al menos con los mecanismos de control de concurrencia en Oracle*9i*, que la ejecución concurrente de sub-transacciones, pertenecientes a la misma transacción o no, una que inserte una entidad en P con A=x y la otra que suprima en R con Kr=x introduzca incoherencias. Por supuesto que hay la posibilidad de abrazo mortal como en todo mecanismo basado en bloqueos.
- Cuando la operación no tiene riesgo de violar la restricción, por ejemplo *Delete* sobre P, o *Insert* sobre R, igual debe realizarse el chequeo; ya que estas operaciones tocan la restricción y podrían ser reparadoras de una inconsistencia introducida por una sub-transaccion ejecutándose en paralelo con ella dentro de la misma transacción distribuida. Por esta razón, se propone verificar la restricción para toda transacción que la toque. Es cierto que cuando no haya concurrencia con otras operaciones podría ser redundante, pero es necesario para garantizar que no se detecten "falsas" violaciones de una restricción.
- Oracle*9i* maneja los locks de escritura distribuidos, esto es los bloqueos de una transacción distribuida son mantenidos hasta el final del two-phase commit en todos los nodos. Esto asegura que la aplicación de nuestro método sobre esa versión funciona correctamente.

2.4. Restricciones Globales Conjuntivas.

Un tipo de restricción más general que la referencial y que expresa relaciones más complejas entre datos de diferentes bases de datos se denomina *restricción global conjuntiva* (ver [2 y 3]). Considerando las bases de datos de la Figura 7 donde existe una relación entre la tabla P en el sitio S_1 y la tabla R en el sitio S_2 a través de la referencia F, una restricción Global Conjuntiva C_f puede ser expresada de la siguiente manera:

$$\forall(p \in P, r \in R),\ p.D = valor_1 \wedge p.\mathrm{F} = r.\mathrm{Kr} \Rightarrow r.\mathrm{N} = valor_2$$

C_f establece que toda tupla en la tabla R relacionada con una tupla en la tabla P (a través de la referencia F) que posea un valor dado para el atributo D, debe a su vez satisfacer una condición determinada para otro atributo N diferente a la clave que las relaciona.

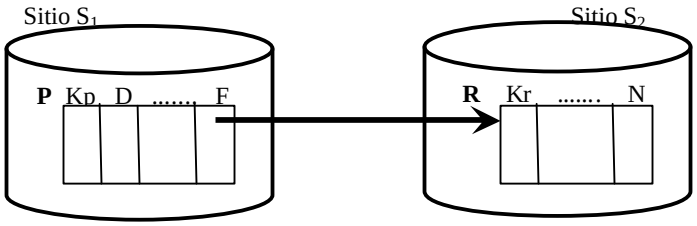


Figura 7: Relación entre P y R a través de la referencia F

La Figura 8 presenta el código pseudo-SQL y la correspondiente transacción anidada que garantiza la restricción C_f cuando se ejecuta una transacción que modifica la tabla P en el nodo S_1. La ejecución de una transacción que modifique la tabla R en S_2 origina la creación de una transacción anidada similar con las funciones invertidas en

cada nodo. La ejecución de dos transacciones en paralelo, una que toque la tabla P en S_1 y la otra la tabla R en S_2, genera una transacción anidada con estructura similar a la mostrada en la Figura 2.

```
-- Check(PL1) en S1
UPDATE INTO P (D) Values ( ),
Intersite: Pupd = SELECT P.F FROM P WHERE D = valor1;

Transfert Pupd from S1 to S2;

-- Check(RG21) en S2
If (SELECT r.Kr FROM Pins p, R r
      WHERE r.Kr=p.F AND r.N ≠ valor2)) ? ∅
Then   ROLLBACK
Else  COMMIT
Endif;
```

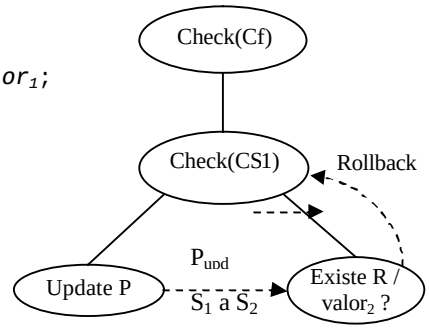


Figura 8: Verificación de restricción global conjuntiva tocada por una transacción que modifica entidades en la tabla P

3. Evaluación del chequeo de restricciones de integridad globales conjuntivas

La experimentación realizada se centra en la verificación de restricciones de integridad globales conjuntivas en transacciones anidadas distribuidas. La medida a comprobar en este ambiente distribuido es: ¿Cuánto tiempo consume realizar el chequeo de una restricción global conjuntiva?. Esta medida es relevante ya que el costo de la transferencia de información entre sitios va a determinar si es factible o no el uso de las soluciones aqui propuestas en MDBS. Para esta experimentación se consideró como contexto una restricción de integridad *C* y se limitó la implementación del Manejador de Transacciones Anidadas Distribuidas al control de la transacción *SCA(C)* y el conjunto de las subtransacciones hojas que tocan *C*, *denotado* T(*C*). Esto significa que no necesariamente se ejecuta una transacción anidada completa, sino un subconjunto de ella que contiene: SCA(C) $\cup$ T(C). Igualmente, en la implementación se simplifica el modelo de transacciones anidadas y se considera que las hojas son ejecutadas siempre en paralelo y se limita el control de concurrencia a los mecanismos provistos en el manejador Oracle*9i* [7].

Para efectos de tomar la medida de interés, en la implementación se descompuso la ejecución de la transacción *SCA(C)* en tres partes o etapas y se midió el tiempo en realizar cada una:

- **Tiempo de Actualización**: Es el tiempo de ejecución de la transacción ó transacciones hojas propiamente dicho, es decir el tiempo de operación sobre los datos y de creación de los correspondientes conjuntos *intersitios*. Este tiempo es tomado en la transacción *SCA(C)* y corresponde al mayor de los tiempos asociados a la ejecución de una transacción hoja.
- **Tiempo de Transferencia**: Tiempo que toma trasmitir un conjunto *intersitio* (en los casos que lo haya) de un nodo a otro. Este tiempo es medido en la transacción *SCA(C)* y corresponde al mayor tiempo asociado a la ejecución de hilos encargados de transmitir los datos.
- **Tiempo de Chequeo**: Esta medida corresponde al tiempo del proceso de verificación de la restricción propiamente dicho. Este tiempo es medido en la transacción *SCA(C)* y corresponde al mayor tiempo de ejecución de los hilos encargados de verificar la restricción. Es importante hacer notar que se consideró el peor caso, es decir la situación en la cual la restricción no es violada y por tanto el proceso de chequeo debe ser realizado para la totalidad de los datos involucrados.
- **Tiempo Total**: Es la suma de los tres tiempos anteriores y representa el tiempo que consume realizar el chequeo de una restricción global conjuntiva en una transacción anidada. Esta es la medida que se toma como métrica para este estudio, en razón de que en cada una de las etapas descritas se toman acciones relacionas con el proceso de chequeo de la restricción.

3.1. Ambiente de Desarrollo.

El hardware utilizado para la implementación del prototipo fue un cluster de 7 sitios homogéneos instalado en el Centro de Computación Paralela y Distribuida (CCPD) de la Universidad Central de Venezuela. Para la experimentación se utilizaron hasta tres sitios del Cluster a la vez.

La especificación de cada sitio es la siguiente: DELL Modelo 4100 Dimension, Procesador Intel Pentium Pro III 1 GHz, 256 MB SDRAM, Disco Duro: 40 GB, Tarjeta de Red 3Com Fast Ethernet 10/100 Mbps. La conectividad

entre los PC se obtiene a través de un Switch 3Com Superstack *3 (3C16980A), Familia 3300 Fast Ethernet de 24 puertos a 10/100 Mbps. Todas las máquinas comparten un monitor, un teclado y un mouse, mediante un Master Console II Raritan de 8 puertos.

El software que se usó en la implementación es el Sistema Operativo Red Hat Linux Advanced Server Release 2.1 AS (Pensacola) Kernel 2.4.9-e.3, el Sistema Manejador de Base de Datos Oracle9*i* Release 2 (9.2.0.1.0), los lenguajes Java 2 Standard Edition J2EE versión 1.4.2, Oracle9*i* PL/SQL, Oracle SQL y Java Database Connectivity JDBC 3.0 API.

3.2. Caso de estudio.

La experimentación se realizó con el ejemplo plasmado en la Figura 9, en donde se presenta una multibase de datos de *Productos* y *Laboratorios* fabricantes de los mismos, sobre la cual se estableció la siguiente restricción: "el fabricante de todo producto de distribución limitada debe tener una clasificación de nivel 33":

$$\forall(p \in \text{Productos}, l \in \text{Laboratorios}),\ p.\text{Distrib} = \text{“Limitada”} \wedge p.\text{Fabricante} = l.\text{Kl} \Rightarrow l.\text{Nivel} = 33$$

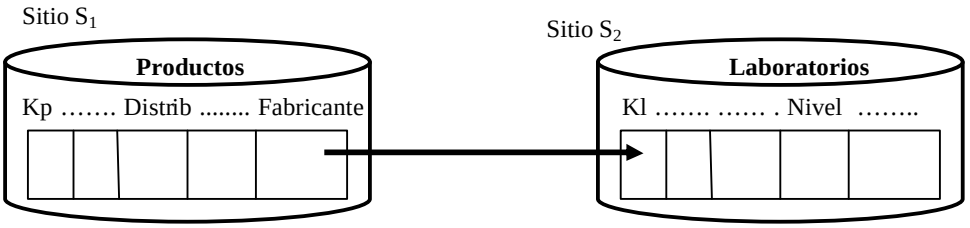


Figura 9: Multi base de datos usada en la experimentación

La cardinalidad de cada tabla se estableció en un millón de registros generados de forma sintética. Aunque el diseño de la BD es bastante sencillo, esto no simplifica el algoritmo implementado para verificar restricciones de integridad. Por otra parte, como lo muestran Grefen y Widom [5] las restricciones que involucran más de dos sitios son muy poco frecuentes y en general las restricciones multisitio pueden ser reformuladas de manera tal de expresarlas en base a múltiples restricciones entre dos sitios.

3.3. Diseño general del prototipo.

Como se muestra en la Figura 10 el prototipo desarrollado para esta experimentación contempla tres (3) capas: el Módulo de Transacciones y Restricciones Globales (MGT/R), el Módulo de Control Local (MCL) y el Módulo de Base de Datos (MBD), siendo el primero la capa superior.

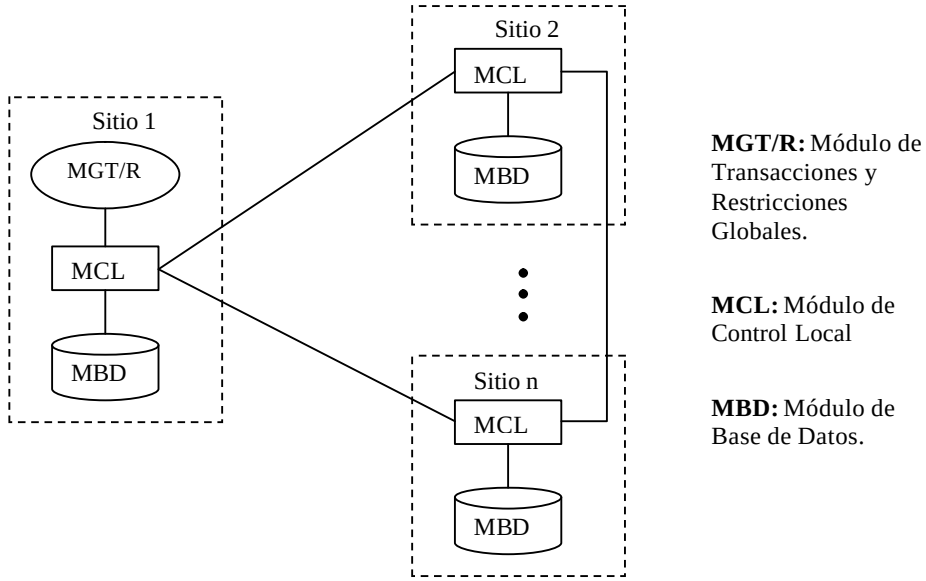


Figura 10. Diseño genérico del prototipo para verificar restricciones de integridad en transacciones distribuidas

El Módulo de Transacciones y Restricciones Globales se encarga del control de ejecución de la transacción *SCA(C)* y de los procesos asociados al chequeo de *C*. Las funciones realizadas por este módulo son: Iniciar la ejecución del prototipo, coordinar la manera cómo se van a ejecutar las operaciones sobre los datos dependiendo de los sitios que toque la transacción, decidir donde y cuando se ejecutan los procedimientos necesarios para el chequeo de la restricción global y decidir si la transacción valida ó no. Dado que en este prototipo se considera la ejecución de una transacción *SCA* a la vez este módulo sólo va a estar presente en el sitio que comience la ejecución, como se muestra en la Figura 10. Este módulo va a establecer una conexión con el Módulo de Control Local presente en el mismo sitio, a través del cual va a ordenar la ejecución de los distintos procedimientos.

El Módulo de Control Local va a estar presente en todos los sitios en los cuales se ejecuta el prototipo y va a tener la capacidad de activar tareas de manipulación de datos, así como la facultad de transmitir tanto datos como órdenes hacia los otros Módulos de Control Local presentes en los demás sitios del sistema (ver Figura 10). Las tareas que va a activar sobre los datos los va a realizar a través de la capa inferior de Base de Datos presente en el mismo sitio y son: ordenar una operación DML (Data Manipulation Language), crear el conjunto intersitio y chequear si los datos del sitio cumplen con la restricción de integridad (verificación del predicado local). La transmisión de datos ocurre cuando se le ordena transmita el *intersitio* al sitio que sea el responsable de chequear la restricción de integridad global.

El Módulo de Base de Datos es la capa encargada de almacenar y gestionar los datos. Este módulo está soportado por Oracle9*i*, heredando por tanto las funcionalidades y facilidades provistas por este manejador [7].

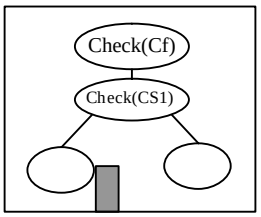
El diseño genérico del prototipo se proyectó a implementaciones particulares en cada uno de los experimentos realizados respetando en todos los casos la arquitectura descrita.

3.4. Experimentos realizados.

Para la evaluación se implementaron cuatro (4) experimentos los cuales fueron definidos en función de las diferentes alternativas para distribuir las dos tablas (un nodo, o dos nodos) y la distribución de las actualizaciones realizadas sobre la base datos, de la transacción global y del proceso de chequeo. A continuación se describe cada experimento realizado:

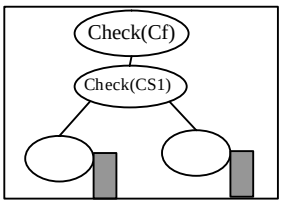
Experimento 1: Un sólo sitio y actualización de una sola tabla

Este experimento es considerado como de control, puesto que esta implementación del prototipo se realizó en un ambiente centralizado. Constituye un escenario para comparar la implementación distribuida con el comportamiento del prototipo en un ambiente centralizado implementado bajo condiciones similares. Todas las tablas están almacenadas en uno de los sitios del clúster y por tanto no hay transferencia de datos de un nodo a otro. Se tiene una transacción hoja que actualiza la tabla *Productos* y la ejecución de hilos se realizó de forma concurrente.



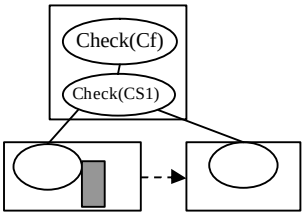
Experimento 2: Un sólo sitio y actualización de ambas tabla.

Este experimento es similar al anterior, excepto que se tienen dos sub-transacciones concurrentes, una que actualiza la tabla *Productos* y la otra la tabla *Laboratorios*.



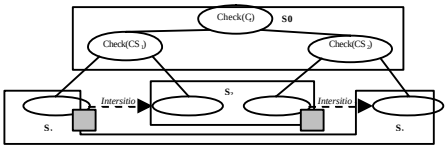
Experimento 3: Tres sitios y transacción actualizando una sola tabla.

Este escenario se implementó utilizando tres sitios del cluster: la transacción $SCA(C_f)$ está ejecutándose en el nodo S_0, la transacción hoja que actualiza *Productos* se ejecuta en el sitio S_1 y la tabla *Laboratorios* está almacenada en el sitio S2. Se requiere la participación de los tres nodos para realizar el proceso de verificación de C_f. El control de ejecución de la transacción y del proceso de chequeo de la restricción es realizado desde el Sitio S_0, donde se llevan a cabo las siguientes funcionalidades: generar la ejecución de las transacciones en los sitios donde están alojados los datos, ordenar la transferencia de datos de un sitio a otro, chequear si la restricción de integridad es respetada y en el caso de que esto suceda,



ordenar la ejecución del mecanismo de validación a dos fases (two phases commit) para que los datos queden consistentes en las BD.

Experimento 4: Tres sitios y transacciones actualizando ambas tablas.
Este experimento es similar al caso anterior, excepto que se tienen dos subtransacciones: una que modifica la tabla *Productos* en S1 y otra *Laboratorios* en S2.



3.5. Tamaño de los experimentos.

Para observar el comportamiento con distintas cargas de datos, para cada experimento se realizaron pruebas con seis (6) conjuntos de datos a ser modificados, cada uno de diferente cardinalidad. Por cada experimento la operación de UPDATE se ejecutó sobre cada una de las tablas, con modificaciones de 20.000, 50.000, 100.000, 200.000, 500.000 y 1.000.000 de registros. Para garantizar confiabilidad en los resultados, por cada conjunto de datos se repitió la prueba diez (10) veces, siendo los tiempos reportados en cada caso el promedio de estas 10 ejecuciones.

Las modificaciones fueron implementadas de manera tal que la restricción se cumpla siempre, es decir, se construyeron casos de prueba que garantizan que la transacción no viole la restricción. Esto porque se quiso evaluar el peor caso en la verificación: el caso donde es necesario chequear todos los datos modificados, obteniendo así el tiempo máximo de ejecución del proceso de chequeo. Si no se hubiese hecho así, el algoritmo se detendría en un tiempo aleatorio dado por el instante en el cual se encuentre el primer elemento que no cumple con la restricción.

4. Resultados Obtenidos

La Figura 11 muestra el Tiempo Total obtenido para los experimentos 1 y 3, en donde se actualiza sólo la tabla *Productos*. Se puede observar que en el experimento 3 el Tiempo Total disminuye ligeramente con respecto al experimento 1, al realizar el chequeo de la restricción en un ambiente distribuido con actualizaciones inferiores a 200.000 y aumenta a partir de esa carga de datos, permaneciendo las curvas de ambos experimentos muy cercanas.

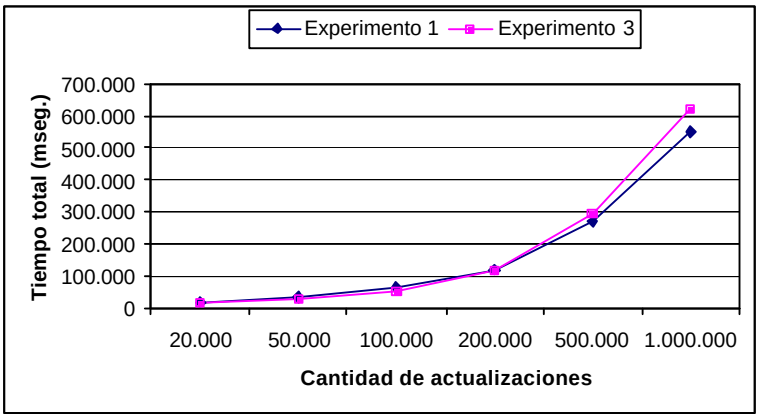


Figura 11. Experimentos de actualización sólo de la tabla Productos

La Figura 12 muestra el Tiempo Total obtenido en los experimentos 2 y 4, cuando la transacción involucra actualizaciones en ambas tablas *Productos* y *Laboratorios*. En este caso se hace evidente que en el experimento 4, a pesar de haber una transmisión de los conjuntos *intersitios* entre los dos nodos, el Tiempo Total es menor que en el experimento 2. Esto se debe a que en el experimento 2 las transacciones están ejecutándose concurrentemente sobre un único nodo realizando la actualización de la BD, generando los intersitios y realizando el chequeo de la restricción. Se observa que el Tiempo Total disminuye, con una aceleración cercana a 2, al pasar del experimento 2 al 4, debido a la paralelización de los procesos de actualización, generación de intersitios y chequeo y la consiguiente reducción de los tiempos asociados, lo cual hace irrelevante el impacto de la transferencia de datos.

Estos resultados muestran que la solución propuesta para verificar restricciones de integridad conjuntiva no degradará el sistema si se implementa en un ambiente distribuido, ya que la posibilidad de ejecución simultánea

anula el efecto negativo que pueda tener la transferencia de datos requerida sobre el tiempo de chequeo de restricciones.

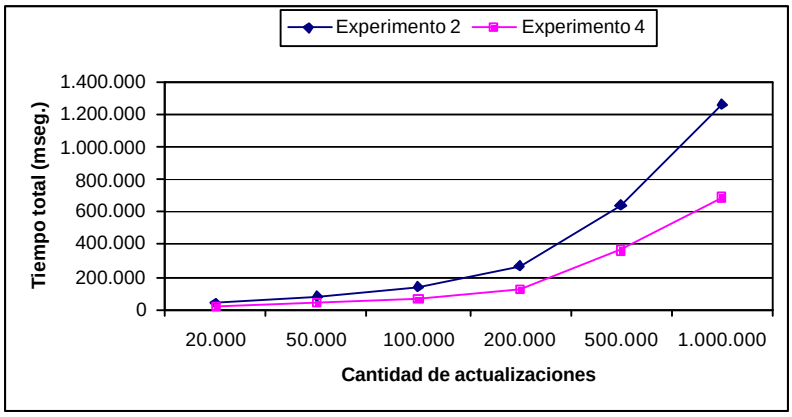


Figura 12. Experimentos de actualización de ambas tablas Productos y Laboratorios

5. Conclusiones

En este artículo se proponen soluciones para mantener restricciones de integridad referencial y restricciones globales conjuntivas en MDBS relacionales. Se analizan las operaciones que pueden violar este tipo de restricciones y se proponen los mecanismos que deben ser introducidos en cada caso para garantizar la integridad de los datos.

Así mismo, se mostró la factibilidad de la solución propuesta para restricciones de integridad globales conjuntivas a través de la implementación de un prototipo de un MDBS sobre un cluster de procesadores homogéneos bajo Linux con Oracle9*i*.

Los resultados muestran que el Tiempo Total para verificar una restricción global conjuntiva con sub-transacciones paralelas que la tocan en dos nodos puede reducirse a la mitad con respecto al caso centralizado, debido a la paralelización de los procesos de actualización y chequeo, lo cual reduce los tiempos asociados haciendo irrelevante la incidencia de la transferencia de datos sobre el Tiempo Total.

La experimentación presentada en este trabajo no considera la ejecución concurrente de múltiples transacciones, ni la verificación de múltiples restricciones tocadas por una misma transacción. Actualmente se está extendiendo el prototipo para realizar experimentos que consideren múltiples transacciones en concurrencia.

Referencias Bibliográficas

[1] J. Akoka and I.Comyn-Wattiau. *Conceptions des Bases de Données Relationnelles* Vuibert. Paris. 20001.

[2] A. Doucet, S. Gançarski, C. León and M. Rukoz. *Integrity Constraints in Multi-DB with Nested Transactions*, LNCS. Volume 2172. Proceedings 9th International Conference of Cooperative Informat ios Systems, COOPSI'2001, Italia, Septiembre 2001, Springer-Verlag.

[3] S. Grufman, F. Samson, S. M. Embury, P. M. D. Gray and T. Risch. *Distributing Semantic Constraints Between Heterogeneous Databases.* Proceedings of the Thirteenth International Conference on Data Engineering, ICDE 1.997, April 7-11, pages 33-42, Birmingham U.K., April 1-997. IEEE Computer Society.

[4] A. Gupta and J. Widom. *Local Verification of Global Integrity Constraints in Distributed Databases.* Proc. of the 1.993 ACM SIGMOD Int. Conf. on Management of Data, volume 22 of ACM SIGMOD Record, pages 49-58, Washington, USA, May 1.993. ACM Press.

[5] P.W.P.J. Grefen and J. Widom. *Protocols for Integrity Constrainns Checking in Federated Database*. Distributed and Parallel Database, 5(4):327-355, Octubre 1997.

[6] J.E.B. Moss. *Nested Transactions: An Approach To Reliable Distributed Computing.* MIT Press, Cambridge, USA, 1.985.

[7] Oracle. *Oracle9i Database Administrator's Guide Release 2 (9.2)*. Part No. A96521-01. Oracle Corporation. March, 2002.

[8] A. P. Sheth and J. A. Larson. *Federated Database Systems for Manageing Distributed, Heterogeneous, and Autonomous Databases.* ACM Computing Surveys, 22(3):183-236, September 1.990.

Best Link Contribution: A Multicast Routing Algorithm Class

Reinaldo C. Vallejos, Alejandra B. Zapata
Universidad Técnica Federico Santa María, Departamento de Electrónica,
Casilla 110-V, Valparaíso, Chile
reinaldo@elo.utfsm.cl, azb@elo.utfsm.cl

Verónica D. Gacitúa
Corporación Nacional del Cobre,
Huerfanos 1270 p.10, Santiago, Chile
vero@ieee.org

Abstract

In this paper a of multipoint routing algorithm class known as "Best Contribution" (BC) is presented. This class of algorithms adapts to changes that may occur in the membership of the group and satisfy the Quality of Service requirements of connections.

To connect new members to the multipoint group, BC expands the distribution tree that connects the current members of the group by adding one link at a time, until all the members of the distribution tree are connected. To select the link to be used for expanding the tree in each step of the algorithm, BC first assigns to each candidate link a value that is given by a specific function, called the "link contribution" function. BC then chooses the link that displays the "best contribution", which becomes part of the distribution tree.

The performance of BC was evaluated by carrying out a set of simulation experiments. They showed that the proposed algorithm performs better than two algorithms that have been identified elsewhere as displaying an excellent performance.

***Index Terms*—**Multicast Routing, Network Communications, Quality of Service.

1. INTRODUCTION

In the early days of computer networks, the point-to-point communication paradigm – one transmitter and one receiver – was adopted from the world of telephony. Examples of this are the transmission of files between computers and the exchange of messages between two users on different machines. This point-to-point transmission has continued to be the dominant paradigm for communication for many years. In the last decade, however, the increase in personal computers processing speed and the success of the Multicast Backbone (Mbone) [1] have made possible the emergence of important applications which allow the exchange of information among a group of users, such as LAN TV, desktop conferencing, corporate broadcasts, and collaborative computing. These are referred to as multicast or multipoint applications.

Multicast applications can be composed a specific transmitter and a destination group (as in audio/video broadcast, and updating of databases), various transmitters and various receptors (as videoconferences, multiplayer games or shared whiteboards) or various transmitters and one receptor (as resource searches or data collectors).

To carry out a multipoint communication, several technical problems related to the correct sending and reception of information within the group must be solved. One of these problems consists on finding the routes to connect the members of the group, so that the information arrives at all the intended destinations at minimum cost, where the cost may be interpreted in different ways in order to optimize different performance measures. This problem is known as *multipoint routing* and is of fundamental importance, due to the fact that choice of the routes impacts significantly on the level of quality of service (QoS) perceived by the final users and the efficiency of the network resource utilization.

The quality of service (*QoS)* of an application refers to the specification of a set of performance measures and the values (or range of values) these measures must achieve in order for the application to operate satisfactorily. For example, QoS constraints may be specified in respect of delays in the transfer of information, the available bandwidth of a link, or the probability that an information packet does not arrive correctly at every one of its destination points.

Guaranteeing the QoS of a connection is not a trivial task for a network. This is due in part to the fact that the connection's demand for resources may vary over time (in the case of connections with variable traffic). Another reason for the difficulty of predicting whether the network can assure the QoS of the connection is that the network's resources (processors, communication channels, etc.) are shared with other connections which may also generate variable traffic over time. The solution to this problem normally proceeds in two stages. First, the connection QoS is translated into a given QoS which must be achieved by each one of the channels making up the connection [2]. (This means that if each channel complies with its QoS, the connection complies with the QoS required by the application). And second, to determine whether or not a given channel complies with the QoS required of it by the connection, a probability study is conducted in which both the characteristics of the channel and those of its users (present and former) are taking into account. One of the most popular techniques for conducting such a study is know as the Bandwidth Equivalent Technique [3], [4]. It consists in translating the connection QoS (e.g., the packet loss probability) into a given bandwidth requirement. This means that if the link assures that this bandwidth may be used by the connection, it complies with the required QoS. Thus, in the rest of this paper we will use without distinction the concepts of QoS required by the connection and the bandwidth needed to serve it.

Given the difficulty of achieving QoS for an application, and the fact that a routing algorithm must use network resources efficiently (objectives which may involve a trade-off), the design of multipoint routing algorithms is a complex problem. Moreover, this complexity is aggravated by the fact that the members of a group may change or the network's topology may be modified (e.g., by failures or repairs in the links or nodes).

The technique initially proposed for resolving this problem of multipoint routing was to communicate every group source with every other member of the group by means of individual point-to-point connections [5], [6]. However this solution wastes network resources [7]. For example, if a given link belongs to every point-to-point connection, each packet must be transmitted through this link as many times as there are receivers in the multipoint group. Also, maintaining separate routes for each different source in the group involves using a high number of network nodes and links to interconnect the group, and imposes a high memory requirement on the routers.

Subsequently, as routers gained multicast capability [10] (that is, the ability to replicate the packet at one input to various outputs), multiple transmission of the same packet was avoided and communication between group members became more efficient. Currently there is a consensus that the most efficient way of connecting members of a group is through a tree known as a distribution tree. This tree is a subset of the graph representing the network, and is composed of the multipoint group nodes and the links and other nodes necessary for connecting them.

In order to build the distribution tree various algorithms have been proposed, which can be classified into three categories: Steiner trees, center-based trees and source-based trees [7].

The algorithms for building source-based trees choose the root of the tree as the group source, and resolve the point-to-point routing problem between the source and each of the group members. Any loops that may have formed in the process of establishing the routes between the source and the group members are eliminated. If the algorithm must satisfy certain QoS constraints, the distribution tree obtained may be refined.

In cases where more than one group member may act as a source, the generation of a different tree for each source would be a waste of resources. To avoid this, a single distribution tree can be created through which any member which needs to operate as a source may transmit information to the rest of the group. To generate this single tree, two different paradigms have been proposed: the Steiner tree and the center-based tree.

The problem of generating Steiner trees existed long before multipoint applications (they were needed for other applications), and consists in finding the lowest-cost tree that interconnects a given subset of nodes of a graph. In the context of multicast routing, the subset of nodes to be interconnected corresponds to the members of the multipoint group. If necessary, some nodes that do not belong to the subset may be included in the tree. These non-members are known as Steiner nodes.

In [8] it was demonstrated that the construction of a Steiner tree is an NP-complete problem, that is, the time needed to find the optimal tree grows exponentially with the size of the graph. For a network with a large quantity of nodes this task would be impossible to execute within a reasonable time, so instead of seeking the optimal solution, good solutions are sought using heuristics. These consist in rapidly constructing a tree whose cost is similar to that of the optimal tree. Many such heuristics for constructing Steiner trees have been proposed in the literature [9], [10], [11], [12], but the majority of them are not suitable for use with the Internet because changes in the network's topology or in the membership of the group require that the heuristic algorithm be re-executed from the beginning. (Hereafter, we shall refer to these as static heuristics.) To get around this problem, dynamic heuristics have been proposed which adapt to changes in group membership, and thus are appropriate for multipoint applications.

The algorithms used to construct center-based trees select one or more network nodes to be information reception and transmission center(s). The selection of centers is also an NP-complete problem [7], for which various center-selection heuristics have been proposed [13], [14], [15]. In group communications through a center-based tree, each group member transmits and receives all information through the center(s). Center-based algorithms seek a

compromise between the delay in the arrival of information from the source to the other group members and the amount of network resources used. The difficulty with this type of tree is that it tends to produce congestion around the centers.

In this study, we propose a multipoint routing algorithm class called "Best Contribution" (BC), that adapts to changes that may occur in group membership and satisfy the QoS requirements for connections.

The rest of this article is structured in the following way: in section **2** the network model and routing problem are defined. In section **3** the proposed algorithm is described. Section **4** the results obtained from simulation experiments to evaluate the proposed algorithm are presented; and finally, the conclusions are set out in section **5**.

2. NETWORK MODEL AND ROUTING PROBLEM

2.1 Network Model

The network is represented by a directed graph $G = (V, E)$, where V is the set of nodes and E (VxV) is the set of directed links connecting the nodes. $|V|$ and $|E|$ are the number of nodes and links, respectively. If n is a node of V, then S_n is a vector which describes the state of the node $n \in V$. In the same way, S_e is the state vector of the link $e = (n_1,n_2) \in E$. The dimension of the state vectors depends on the number of considered parameters. Examples of parameters for a link can be its bandwidth, the cost of its use, or its propagation delay. The global state of the network corresponds to the set of state vectors of all of the links and nodes of the network.

$M \subseteq V$ is the set of nodes that participate in the multipoint communication, called *group*. The packets originated at a determined source node n_s – which may or may not belong to the group – are sent to the set of node receptors $M - \{n_s\}$, known as destinations. There can be more than one source for one multicast group, and a member of the group can be a source or a destination at different times of the multicast session.

A multipoint tree T is a sub-graph of G that connects all the members of the group and other non member nodes that are necessary to establish the communication.

Consider two non negative valued functions associated with each link $e \in E$: $D(e)$ and $C(e)$. The *delay of the link D(e)*, with $D : E \to \Re^+$, is the average delay that data packets experience once they go through a given link *e*. This is the sum of the times related to switching, queuing, transmitting, and propagating. The *cost of the link C(e)*, with $C : E \to \Re^+$, is a measure of the use of the bandwidth in the corresponding link. The higher the amount of bandwidth used, the higher the cost. It is assumed that the links in a network are asymmetrical, therefore the cost and delay of link $e = (n_1, n_2)$ is different from those of link $e' = (n_2, n_1)$.

The path composed by links and nodes of T that starts at the source node n_s and ends at the destination node $n_d \in M$, is denoted as $P_{T(n_s,n_d)}$. The *cost of the path* $P(n_s, n_d)$ is defined in the following way:

$$C(P(n_s, n_d)) = \sum_{e \in P(n_s,n_d)} C(e) \qquad (1)$$

In the same way, the *end to end delay* (or *source to destination delay*) *of the path* $P(n_s, n_d)$ is defined as:

$$D(P(n_s, n_d)) = \sum_{e \in P(n_s,n_d)} D(e) \qquad (2)$$

The *cost of the tree T* is the sum of all the costs of its links:

$$C(T) = \sum_{e \in T} C(e) \qquad (3)$$

2.2 Multicast Routing Problem

A multicast routing algorithm aims to create a distribution tree T which connects all the members of a multicast group M in an optimal way, according to the objective functions O whilst considering the set of constraints C. The optimization objectives are defined as the minimization (or maximization) of a function of the parameters associated to nodes or links of the multipoint tree (e.g., the cost of the tree, or the average delay between the source and the receivers). The constraints are usually related to the quality of service offered to the destination nodes. For example, for time-sensitive applications the maximum end-to-end delay cannot exceed 100 ms [21]. Hence, the tree must be built so this constraint is not violated.

The resulting distribution tree may vary depending on the objective and restriction functions; the most used restriction functions are described below.

Shortest path trees without restrictions. The aim is to minimize (1), for each $n_d \in M$, using the following objective function:

$$\min_{\forall P(n_s,n_d)} C(P(n_s,n_d)), \qquad n_d \in M \tag{4}$$

Then *T* is constructed as the union of all the selected paths, that is:

$$T = \bigcup_{\forall n_d \in M} P(n_s,n_d)$$

Note that the tree structure depends on the link metrics that are taken into consideration.

Steiner Trees without Restrictions. The aim is to minimize (3), using the following objective function:

$$\min_{T\in\tau} C(T) \tag{5}$$

Where τ is the set of all possible trees that interconnect the set of nodes $\{n_s\} \cup M$.

Shortest path trees with delay restrictions. Similar to Shortest path trees without restrictions, but in this case the path between the source and each member group must satisfy a delay restriction, as follows:

$$D(P(n_s,n_d)) \le \Delta, \quad \forall n_d \in M, \quad P(n_s,n_d) \in T \tag{6}$$

where Δ is the delay limit for all the paths connecting source-destination pairs.

Steiner Trees with delay restrictions. The aim is to minimize (3), using the objective function (5) whilst meeting the restriction (6).

3. Description of the Proposed Algorithm

3.1 Definitions

M is the set of member nodes that belong to the graph *G* that represents the network. *T* is a partial tree that connects one or more members of the group. If *b= (v,w)* is a link such that node $v \in T$ and node $w \notin T$, *then b is a frontier link. B* $\subseteq E$ is the set of *Frontier links*. *Frontier nodes* are those connected to a frontier link, but not belonging to *T*. *U* is the set of members not connected yet to *T*.

The contribution *c(b)* of the link $b \in B$ to connect *U* to *T* is defined as follows:

$$c(b) = \sum_{u\in U} \frac{1}{f(b,u)} \tag{7}$$

Where *f*(*b*, *u*) is a function associated to the path that begins in the frontier link *b* and that bring *T* closer to the non-connected member $u \in U$. The link chosen to expand the partial distribution tree, denoted by $b^* \in B$, must then comply with the following condition:

$$c(b^*) = \max_{b\in B}\{c(b)\} \tag{8}$$

Each term *1/f(b,u)* in (7) corresponds to the contribution that the link *b* makes when it tries to approach the node $u \in U$ to the partial tree *T* and it can be adjusted according to different metrics, as shown in the following three examples:

$\boldsymbol{f_1(b, u)} = 1 + d(adj(b), u)$, where *adj*(*b*) is the frontier node that is connected to *T* by means of a frontier link $b \in B$; and *d*(*adj*(*b*), *u*) is the distance, measured by the number of hops, in the shortest path that begins at the frontier node *adj(b)* and ends at the node $u \in U$. It aims to minimize the number of hops required to reach all the unconnected members.

$\boldsymbol{f_2(b, u)} = 1 + \mathrm{BW}(adj(b), u)$, where BW(*adj*(*b*), *u*) is the sum of the used bandwidth in each link that make up the shortest path that begins in the frontier node *adj*(*b*) and ends in the node $u \in U$. It aims to minimize the bandwidth used by the links required to reach all the unconnected members

$\boldsymbol{f_3(b, u)} = (1 + d(adj(b), u))\cdot \mathrm{BW}(b)$, where BW(*b*) is the is the used bandwidth in the frontier link *b*.

Note that when BC uses objective functions like $f_1(b, u)$ or $f_3(b, u)$, only requires to use local information to operate (found in the unicast routing tables). Instead, if BC uses an objective function like $f_2(b, u)$, it requires global information about bandwidth utilization in all the links of the network. Its objective is to minimize the number of hops required to reach all the unconnected members, but this value is pondered by the bandwidth used by the frontier link *b*.

3.2 Operation

The BC operation can be described in three steps:

1. Randomly select a member of the group. This node constitutes the initial partial tree.
2. Select a frontier link which achieves the best contribution, as defined in (7). Ties are broken randomly. Repeat until all of the members of the group are part of the distribution tree.
3. Prune leaf nodes which are not members of the group.

Step 3 takes care of an infrequent situation which occurs if the algorithm expands the tree through a path that later results inconvenient due to insufficient available bandwidth in the frontier links. In these cases, the algorithm changes the tree expansion direction and leaves some useless leaf nodes.

In a large network with a multicast group relatively small and dispersed within the network there are more frontier links than in a small networks with denser groups. Therefore, in the former case BC tends to perform more iterations than in the latter. However, through many experiments we have observed that -in most cases- once a frontier link is chosen, the next link to be added is one of the adjacent ones to the just added node. Based on this observation, BC heuristic is simplified by evaluating only those frontier links adjacent to the last node included into the partial tree, only if this node is not a member. If this node is a destination (member), all the frontier links are evaluated. We refer to this variant of BC as Best Contribution Light (BCL). BCL operates in the same way as BC but step 3 is not required. In the infrequent cases where the tree cannot be expanded trough a yet initiated path, then that path is eliminated and the expansion begins again trough the state it had when the last member was connected. BCL is highly attractive because it speeds up the creation of the distribution tree (due to the search of a smaller frontier link space) whilst generating trees just slightly worse than those created by BC, as shown in section 4. Pseudo code of the BCL heuristic shows in Fig. 1.

INPUT:
$G(V, E)$: graph that represent the network;
k : any particular node that belongs to M;
M : set of destination nodes;
T_j : partial tree in the j-iteration;
Ady_{ij} : adjacent node to $n_i(T_j)$;
$n_i(T_j)$: node i that belongs to T_j. Note that i can assume values between 0 and the total number of nodes of T_j ;
T_f : final distribution tree that corresponds to T_j in the last iteration of the algorithm.
$Contr(Ady_{ij})$: Ady_{ij} contribution;
TheBest : node connected through the frontier link that has the most contribution in the j-iteration;
GreaterContr : value of the TheBest contribution;
Max : contribution of the node Ady_{ij} when it is unreachable from $n_i(T_j)$, (for example: saturation of the links).
FUNCTIONS
EVALCONTRIBUTION : is a function defined by HEURISTIC that evaluates the contribution of any given node. (HEURISTIC is defines by the objective function that is used to generate the distribution tree);
OUTPUT:
Distribution tree (T_f) that reaches all of the nodes that belong to M.
$j \leftarrow 0$;
$T_0 \leftarrow k$;
GreaterContr $\leftarrow 0$;
AllDest $\leftarrow 0$;
Terminated = False;
DifBC = False;

```
while (Terminated = False ) {
 do {
  while (Ady_ij ≠ NULL) {
    if (Ady_ij ∈ T_j) then
       i++;
    end if
    else
       Contr(Ady_ij) ← EVALCONTRIBUTION(Ady_ij);
       if (GreaterContr <= Contr(Ady_ij)) then
          GreaterContr ← Contr(Ady_ij);
          TheBest ← Ady_ij
       end if
    end else
    i++;
  }
  if ((TheBest ∉ M)&(Ady_ij ∈ T_j, ∀ n_i(T_j)) then
     DifBC = True;
  end if;
  j++;
 } while ((n_i(T_j) ≠ NULL)&(DifBC = False));
 if (Contr(TheBest) = Max) then
    return("The distribution tree could not be built. ☹");
 end if;
 T_j+1 ← T_j + {TheBest};
 GreaterContr ← 0;
 if (TheBest ∈ M) then
    AllDest++;
    if (AllDest = #M) then
       Terminated = True;
    end if;
 end if;
}
return(T_f);
```

Fig. 1 BCL Pseudocode

4 PERFORMANCE EVALUATION

4.1 Simulation Tool

The MCRSIM software [22], was specially develop for the performance evaluation of multicast routing algorithms. MCRSIM develop in North Carolina State University was used as the simulation tool in [20], [30].

The simulation tool was adapted to incorporate the BC algorithms class propose in this article.

4.2 Evaluated Algorithms

Among the multicast routing algorithms proposed to date, – mentioned in surveys [25], [26], [27], [28] – , we compared BC and BCL to the following:

- KMB [10], shown [16] the best algorithm to construct non-constrained Steiner trees.
- BSMA, proved to be the best choice for delay restricted Steiner trees [18].

- KPP [19] and CAO [17], frequently mentioned in the literature (e.g., [25], [26], [27]).
- PRUNED [16], expected to achieve best results when the group is close to the network size
- PIM [29], representing the algorithm associated to the currently implemented multicast routing protocols
- QDMR [20], representing an algorithm that can manage adaptively the construction of the tree for assurance not to surpass the limit delay established as the restriction of the tree.

The following five variants of BC were evaluated, each associated to a different objective function:

- BC-SAL. Uses the $f_1(b, o)$ function described in 3
- BC-BW. Uses the $f_2(b, o)$ function described in 3.
- BC-BW-SAL. Uses the $f_3(b, o)$ function described in 3.
- BCL-SAL. The light version of BC-SAL.
- BCL-BW-SAL. The light version of BC-BW-SAL

Most of the algorithms considered for the evaluation are available in the simulation tool mentioned in 4. These algorithms are: KMB, PRUNED, KPP, CAO, BSMA, and an algorithm associated to PIM-DM protocol [23]. Additionally, the code of QDMR algorithm – which was developed at Northeastern Boston University – was written to operate in MCRSIM. Finally the five different heuristics of the BC family, mentioned previously, were programmed for the simulation tool used in this work.

4.3 Performance Metrics

The following three metrics have been chosen to evaluate the performance of the multicast routing algorithms:

4.3.1. Tree Cost

The cost *C(T)* of the *T* tree measures the algorithm efficiency on the use of network resources. *C(T)* was defined in equation (3) as the sum of the individual link costs that make up the tree. The cost *C(e)* of each link is considered equivalent to the bandwidth reserved in each link, like in [16], [30].

4.3.2 Average Delay from the Source to the Destinations

This metric corresponds to the average of the end-to-end delay of the individual paths from the source to each destination, defined in equation (2). This measure gives some insight in to the generated tree's ability to comply with the delay requirements.

4.3.3 Number of Successful Connections

This metric corresponds to the ratio between the number of accepted connections and the total number of requested connections [16], [30]. A connection is successful when the capacity requirement to build the tree does not exceed the available capacity. For algorithms including end-to-end delay restrictions, a tree is successfully generated if also the end-to-end delay constraint is not violated.

4.4 Configuration of the Simulation Experiments

Two types of simulation experiments were realized. The first one focused on cost and delay metrics whilst the second type was aimed to evaluated the number of successful connections.

4.4.1 Type I experiments

These simulation experiments evaluated, for each algorithm, the cost of the tree and the average delay between source and destinations, in relation with the size of the multicast group. Each executed experiment considered a random network, group, and multicast source. Once the experiment was established, all algorithms were evaluated.

For each algorithm, a multicast session was created. The algorithm builds a distribution tree reserving the required bandwidth in the corresponding links. The session is then deactivated and a new session is requested. The experiment ran repeatedly until the metrics reached a confidence interval of 95%.

For each observation a random Waxman graph was generated [16], [31]. The number of network nodes was varied between 20 and 100 to represent different sized networks. As in [30], the link capacity was assumed to be 155 Mbps and were considered saturated when its use reached 85% of its capacity. For algorithms that consider delay restrictions, the established limit for the average end-to-end delay was 0.03 seconds.

To represent the traffic existent in the network, background traffic with three different network loads was added. These background traffics were uniformly distributed between: 5 and 125 Mbps to represent highly unbalanced networks; 45 and 85 Mbps to represent mediumly unbalanced networks; and 100 and 120 Mbps to represent close to saturation networks.

4.4.2 Type II experiments

This kind of experiment begins with an empty network that is progressively loaded with new multicast sessions, until it reaches 3000 requests. Then the network is emptied and the experiment is repeated until the measures reach a confidence interval of 95%. The same network topology is used during all of the experiments. For each connection request, the network is loaded with a new multicast session, choosing a new random group and source each time.

The chosen network is a Waxman random network with the same parameters as in the first experiment. The number of successful requests is the performance metric under study (a request can be blocked if the delay limit is violated for algorithms with delay restrictions or if links do not have sufficient available bandwidth). Notice that a network service providers would rather implement an algorithm that allows them to accept a larger quantity of connection requests.

4.5 Simulation Results

In this subsection, a description of the results obtained for the different algorithms are made for every metric under evaluation.

4.5.1 Cost of the Tree

Figures 2 to 5 – obtained from Type I simulation experiments – show the relative cost of the distribution tree obtained with each algorithm in relation to the cost obtained by the KMB algorithm The reason for showing the performance of the different algorithms relative to that of KMB, is that this last algorithm has been reported as having one of the best performances.

Figures 2 to 5 show that BC algorithm achieves lower cost than KMB - the best performing algorithm in terms of cost to date- for all the studied cases of traffic load and network size. As expected, among the five BC variants studied, BC-BW-SAL performs the best (because it uses the f_3 function of section 3.1, whit an objective to achieve low cost trees) with a cost reduction of up to 48% with respect to KMB. BCL-BW-SAL also clearly outperforms the KMB algorithm while the remaining variants of the BC family perform close to KMB. None of the remaining studied algorithms were able to achieve cost reduction with respect to KMB. This is a remarkable result, given that KMB has been considered the best heuristic to generate low cost trees during the last decades and as such, it has been often used as a benchmark to compare multicast routing algorithm performance.

The highest cost reduction of BC is achieved when the group size is small compared to the network size, which is precisely expected to occur in practice (can be seen with real world multicast monitoring tools such as [24]. It can also be seen that BCL performance is just slightly worse than BC´s, but given that this is achieved at reduced computational complexity, BCL becomes an attractive alternative for implementation.

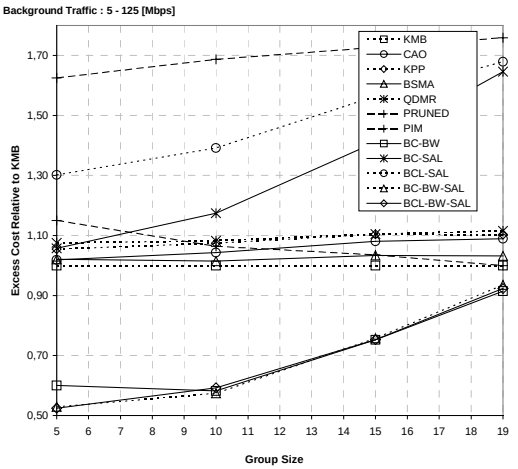


Fig. 2 Cost of distribution tree relative to KMB as a function of the group size. Networks of 20 nodes and background traffic between 5 and 125 Mbps.

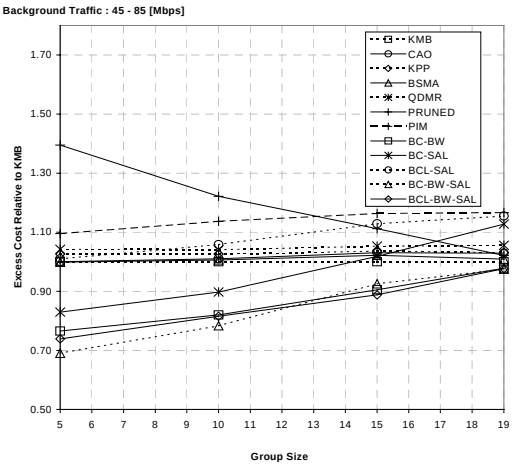


Fig. 3 Cost of distribution tree relative to KMB as a function of the group size. Network of 20 nodes and background traffic between 45 and 85 Mbps.

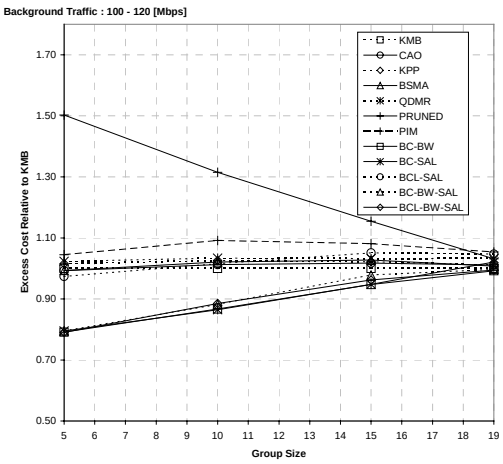


Fig. 4 Cost of distribution tree relative to KMB as a function of the group size. Network of 20 nodes and background traffic between 100 and 120 Mbps.

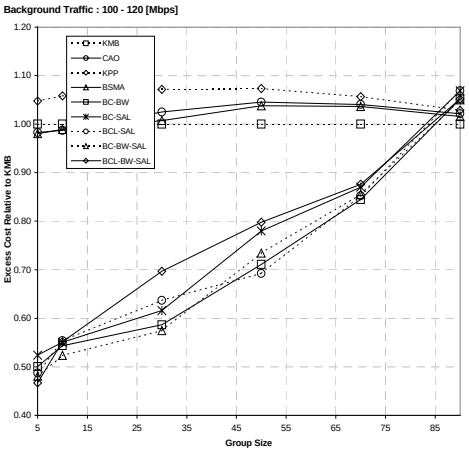


Fig. 5 Cost of distribution tree relative to KMB as a function of the group size. Network of 100 nodes and background traffic between 100 and 120 Mbps.

4.5.2 Average Delay between Source and Destinations

Figures 6 to 9 show the average delay between source and destinations (in seconds). The network scenarios are the same as those indicated in Figures 2 to 5, respectively.

It can be seen that BC-SAL and BC-BW-SAL achieve the lowest delays (and under the established limit) for a wide range of loads and group sizes, with BC-BW-SAL outperforming BC-SAL for small group sizes. The QDMR algorithm also performs good, but in the unlike situations where the group size is near to the network size. However, it must be recalled that BC-SAL achieves this low delay at lower cost than QDMR (See Figures 2 to 5) and hence, it becomes a better option. Besides, the achieved delay is up to 25% under the value achieved by BSMA –the algorithm build to obtain delay-constrained multicast trees.

BC-BW-SAL and BCL-BW-SAL perform similarly and significantly worse that BC-SAL or BCL-SAL for group sizes larger than 50% of the network size. For group sizes smaller than 20% of the network size instead, delay for BC-BW-SAL and BCL-BW-SAL is lower than BC-SAL and BCL-SAL.

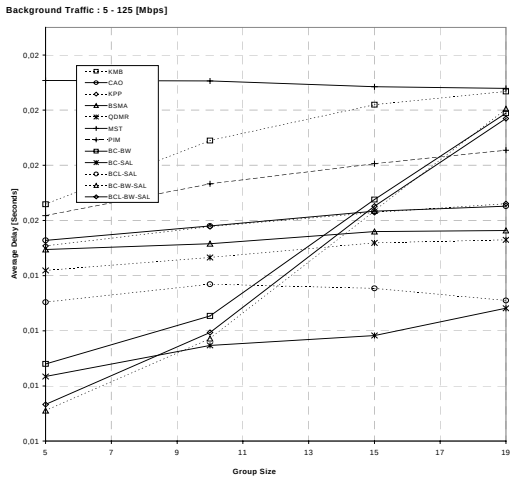


Fig. 6 Average Delay in second. Network of 20 nodes and background traffic between 5 and 125 Mbps.

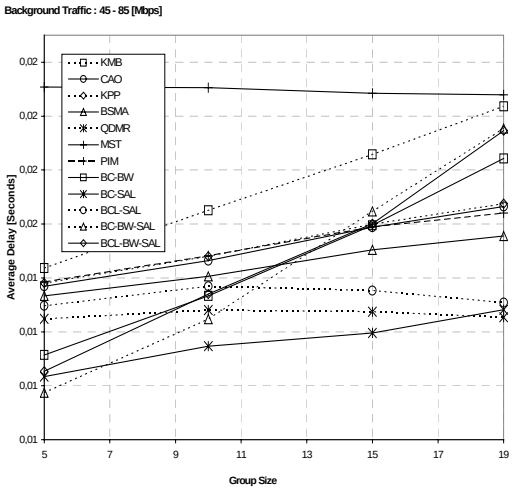


Fig. 7 Average Delay in seconds. Network of 20 nodes and background traffic between 45 and 85 Mbps.

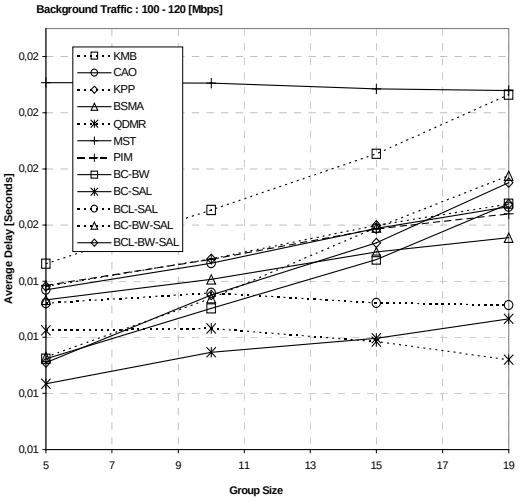


Fig. 8 Average Delay in seconds. Network of 20 nodes and background traffic between 100 and 120 Mbps.

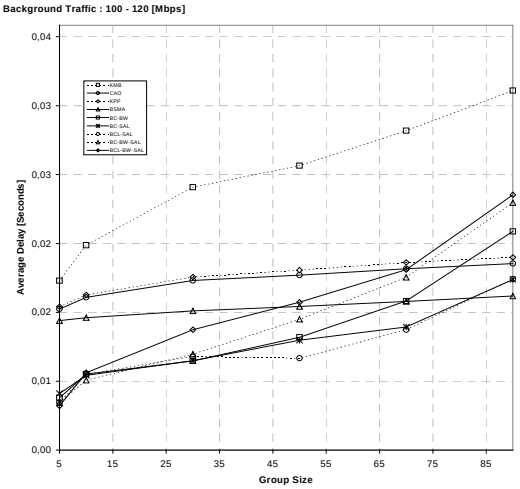


Fig. 9 Average Delay in seconds. Network of 100 nodes and background traffic between 100 and 120 Mbps.

Notice that although BC-BW is not well *tuned* to the evaluated metric (it aims is to minimize the total bandwidth of the tree), its delay is still lower than non-BC heuristics for many cases. This is due to the common aim of BC algorithms of building trees with a small number of links.

4.5.3 Number of Successful Connections

Figs. 10 and 11 show that BC algorithms obtain the best results, accepting over 50% more requests than KMB and KPP, for group sizes smaller than 50% of the network nodes. BC-BW-SAL achieved the best performance among all. For groups with a higher node count the BC algorithms still accept more connections but the difference reduces significantly.

The rest of the evaluated algorithms – BSMA, KPP, KMB, and CAO- accepted a smaller number of multicast sessions per number of requests than the BC family's heuristics.

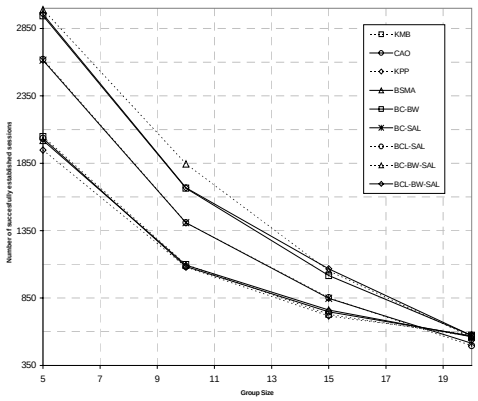


Fig. 10 Number of Successful Connections. Network of 20 nodes, background traffic between 5 and 125 Mbps and total number of requests equal to 3000.

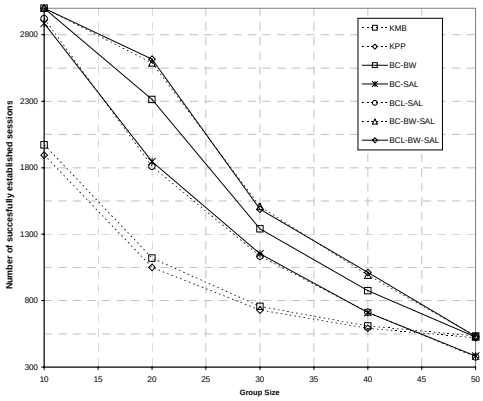


Fig. 11 Number of Successful Connections. Network of 100 nodes, background traffic between 5 and 125 Mbps and total number of requests equal to 3000.

5 CONCLUSIONS

In this paper a multipoint routing algorithms family (BC) has been proposed in which the concept of Best Contribution is employed. The philosophy behind the algorithm consists of expanding the distribution tree one link at a time. The chosen link is that which makes the best contribution in bringing the partial tree simultaneously closer to all non-connected members. This manner of selecting a link is original in two important ways. First, no algorithm yet proposed which operates by expanding the tree attempts to simultaneously connect all non-connected members; rather, they all try to connect one non-connected member at a time. And second, BC in practice corresponds to a "family" of multipoint routing algorithms, since for each function chosen to evaluate the contribution of the boundary links, an algorithm is synthesized which behaves in its own manner. This enables BC to be adapted to different conditions, or to be "tuned" in order to improve its performance as indicated by various measures.

Another notable characteristic of BC is that it operates dynamically, that is, it allows changes to be made in the membership of the multipoint group (a significant number of heuristics proposed as a solution to the Steiner tree do not have this characteristic). Additionally, BC permits various members to connect to the partial tree simultaneously without altering the algorithm's operating philosophy.

Also significant is that BC may be used as a multipoint or point-to-point routing algorithm. In the latter case, the partial tree is made up of one node and only one member is to be connected to it, using the Best Contribution criteria.

The results obtained in the simulation experiments allow us to conclude that BC presents a good performance. For example, on the cost of generating a tree, generally BC outperformed KMB and BSMA, which are considered as one of the best algorithms in the technical literature.

ACKNOWLEDGMENT

Financial support from Fondecyt Project Nº1000055/2000 and USM Project Nº23.02.23 is gratefully acknowledged.

REFERENCES

[1] H. Ericksson, "Mbone. The Multicast Backbone". Communications of the ACM, August 1994.
[2] D.H. Lorenz, A. Orda, "QoS Routing in Networks with Uncertain Parameters", IEEE/ACM Transactions on Networking, Vol.6, No.6, December 1998.
[3] E. Gelenbe, X. Mang, R. Onvural, "Bandwidth Allocation and Call Admission Control in High-Speed Networks", IEEE Communications Magazine, May 1997.
[4] G. De Veciana, G. Kesidis, J. Walrand, "Resource Management in Wide Area ATM Networks Using Effective Bandwidths", IEEE JSAC, Vol.13, No.6, August 1995
[5] Y. K. Dalai, R. M. Metcalfe, "Reverse Path Forwarding of Broadcast Packets", Commun. Ass. Cornput. Mach., vol. 21, Dec. 1978.
[6] K. Kumar, J. Jaffe, "Routing to Multiple Destinations in Computer Networks", IEEE Transactions on Communications, pp. 31:343--351, 1983.
[7] C. Diot, W. Dabbous, J. Crowcroft, "Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms", IEEE Journal on Selected Areas in Communications, vol. 15, nº13, April 1997.
[8] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: freeman, June 1988.
[9] H. Takahashi, A. Matsuyama, "An approximate solution for the Steiner problem in graphs", Math, Japonica 6, pp.573-577, 1980.
[10] L. Kuo, G. Markowsky, L. Berman, "A Fast Algorithm for Steiner Trees", Acta Informatica 15, pp. 141 – 145, 1981.
[11] P. Winter, J. MacGregor S., "Path-Distance Heuristics for the Steiner Problem in Undirected Networks". Algorithmica, Voln 7, Nº2-3, pp.309-327, 1992.
[12] C.A. Noronha, F. Tobagi, "Optimum routing for multicast streams", in IEEE INFOCOM'94, Vol.2, Toronto, Canada, pp. 865-873, June 1994.
[13] A.J. Ballardie, "A new approach to multicast communication in a datagram internetwork", Ph.D. dissertation, Univ. College of London, May 1995.
[14] Thaler, Ravishankar, "Distributed Center-Location Algorithms", IEEE JSAC, Vol.15, No3, 1997.

[15] L. Wei, D. Estrin, “A comparison of multicast trees and algorithms”, Comput. Sci. Dept., Univ. Southern California, Tech. Rep. USC-CS-93-560, Sept. 1993.
[16] H. Salama "Multicast Routing for Real-Time Communication on High-Speed Networks", Ph.D. thesis, Department of Electrical and Computer Engineering, N.C. State University, November 1996.
[17] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real--time Channels." Technical Report (TR-94-024), International Computer Science Institute, UC Berkeley, 1994.
[18] Q. Zhu, M. Parsa, J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting", Proceeding of IEEE INFOCOM'95, pp. 337 - 385, 1995.
[19] V. Kompella, J. Pasquale, G. Polyzos, “Multicast Routing for Multimedia Communication”, IEEE/ACM Transactions on Networking, vol.1, no. 3, Jun 1993.
[20] L. Guo, I. Matta, "QDMR: An Efficient QoS Dependent Multicast Routing Algorithm", Proceeding of 5 th IEEE realtime technology and application symposium (RTAS '99), Vancouver, Canada, June 1999.
[21] N. Seitz, “ITU-T QoS Standards for IP-Based Networks,” IEEE Commun. Mag., vol. 41 (6), June 2003, pp. 82-89
[22] H. Salama, “MCRSIM: The Multicast Routing Simulator. Version 2”, http://rtcomm.csc.ncsu.edu/qos.htm, released September 1997.
[23] A. Adams, J. Nicholas, W. Siadak, “Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)”, Draft IETF, Feb 2003.
[24] http://imj.ucsb.edu/mantra/
[25] B. Wang, J. Hou, “Multicast Routing and Its Extension: Problems, algorithms, and Protocols”, IEEE Network, Jan/Feb 2000.
[26] P. Pragyansmita, S. V. Raghavan, “Survey of multicast routing algorithms and protocols”, Proceedings of the 15th international conference on Computer communication, 2002. Transactions on Networking, Vol.6(6), pp.828-837, December 1998.
[27] S. Chen, K. Nahrstedt "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions", IEEE Network, Special Issue on Transmission and Distribution of Digital Video, Nov./Dec. 1998.
[28] L. Sahasrabuddhe , B. Mukherjee, “Multicast Routing Algorithms and Protocols: A Tutorial”, IEEE Network, Jan/Feb 2000.
[29] A. Adams, J. Nicholas, W. Siadak, “Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)”, Draft IETF, Feb 2003.
[30] H. Salama, D. Reeves, Y. Viniotis. “Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks”. IEEE Journal on Selected Areas in Communications, 15(3):332-345, (p 77), April 1997.
[31] B. Waxman. “Routing of Multipoint Connections”. IEEE Journal on Selected Areas in Communications, 6(9):1671--1622, 1988.

HuyaWeb: Un motor de búsqueda para imágenes basado en contenido

Robinson S. Rivas–Suárez, Nilka Toro, Eduardo Muñoz
Universidad Central de Venezuela,
Escuela de Computación
Centro de Computación Paralela y Distribuida
Caracas 1040A, Venezuela
(rrivas,ntoro,emunoz)@ccpd.ciens.ucv.ve

Abstract

There are many commercial systems that search images on the web, based on meta-data textual patterns. Some of such systems, as Google, Altavista and Yahoo! performs complex operations based not only on the image name, but also in the presence/absence of some key words in the pages where the images are allocated. However, none of the most popular systems do Content Based Image Retrieval.

Some experimental/academic works in progress try to improve the textual image search, adding some degree of CBIR. In this work we present HuyaWeb, a web-based search engine specifically designed for images, where the search criteria are the images features. We used the Query-by-example paradigm, so users provide a query image and the system retrieves the most similar images based on pictorial features. No textual data is required.

Result images can be obtained both from the engine's database or directly from the web, searching initially from a web site provided by the user. We show some examples of usage and discuss results and further works.

Keywords: Content-Based Image Retrieval, Web Search Engines

Resumen

Actualmente existen muchos sistemas comerciales que permiten buscar imágenes en el Web basándose en información textual. Algunos de esos sistemas, como Google, Altavista o Yahoo! realizan complejas comparaciones para ubicar las imágenes basándose no solamente en el nombre de la imagen, sino también en la presencia de las palabras clave dentro de las páginas o contextos donde se encuentra la imagen. Sin embargo, ninguno de los sistemas más utilizados hace Recuperación de Imágenes Basada en Contenido (CBIR).

Algunos sistemas experimentales y académicos tratan de mejorar las búsquedas basadas en texto añadiendo alguna forma de CBIR. En este trabajo se presenta HuyaWeb, un motor de búsqueda para el web diseãdo específicamente para imágenes, donde el criterio de búsqueda son las características de las propias imágenes. Se usa el paradigma de Búsqueda por Muestra, de manera que los usuarios proveen una imagen de muestra y el sistema retorna las imágenes más similares de acuerdo a las características pictóricas. No se utiliza ninguna información textual.

Las imágenes resultado pueden obtenerse de la Base de Datos del sistema, o directamente desde el web. En este caso, las búsquedas comienzan en un site provisto por el usuario. Se muestran ejemplos de uso del sistema y se discuten los resultados y trabajos futuros.

Palabras claves: Recuperación de Imágenes Basada en Contenido, Motor de Búsqueda

1 Introducción

Buscar información en el Web no es una tarea sencilla. En general, las técnicas más avanzadas se han desarrollado para búsquedas de documentos de texto. Sin embargo el aumento de la cantidad de información disponible en el Web hace necesario el desarrollo de técnicas y herramientas para indexar y recuperar otro tipo de información no textual, tal como imágenes, audio, videos, presentaciones, etc.

En particular con imágenes, las dificultades con respecto a la adecuada indexación y recuperación comienzan con la definición del concepto de similitud. Así, mientras la búsqueda de una cadena de texto dentro de un documento es una operación conocida, comparar y buscar imágenes de acuerdo a sus características sigue siendo una tarea compleja y no resuelta del todo.

Para la búsqueda de imágenes digitales en el Web, muchos de los motores de búsqueda existentes actualmente como Google, Yahoo, etc. utilizan la información textual presente en el entorno de la imagen como palabras claves en la página y nombre del archivo. En base a esta información deciden si tomar la imagen o no. Estos sistemas sin embargo no poseen información con respecto a las características de las imágenes, lo que es desventajoso al momento de presentar resultados a los usuarios. En especial los motores de búsqueda comerciales actuales no pueden buscar imágenes en base a una imagen de referencia, o de acuerdo a características como color, forma o presencia de objetos.

En este trabajo se presenta una alternativa para la búsqueda de imágenes en el Web, mediante el diseño e implementación de un motor de búsqueda para imágenes digitales basado en contenido. Esto es, dada una imagen de referencia, se desarrolló un sistema que permite buscar imágenes similares a aquella en el Web, usando únicamente las características pictóricas de la imagen.

Para este trabajo se diseñaron e implementaron cuatro módulos:

- **Cliente:** permite a los usuarios la recuperación de imágenes basado en contenido (CBIR), mediante la especificación de la imagen de referencia.
- **Crawler:** permite buscar e indexar recursivamente todas las imágenes que se encuentren en un determinado sitio Web, repitiendo recursivamente las búsquedas, hasta satisfacer un criterio de parada.
- **Aplicación Servidor:** permite procesar las solicitudes de imágenes de los clientes y proveer las respuestas según los requerimientos solicitados.
- **Base de Datos:** Almacena eficientemente toda la información de imágenes y el URL de cada una de las páginas Web indexadas y recuperadas por el Crawler.

El sistema desarrollado se encuentra en etapa experimental. Los algoritmos utilizados para la indexación y extracción de características de las imágenes solamente toman en cuenta aspectos pictóricos relacionados al color, y no se toman en cuenta descriptores más especializados y complejos como descriptores de formas o texturas. Sin embargo, la arquitectura del sistema y la implementacioón modular permiten añadir funcionalidades de manera sencilla, y adaptada a las necesidades del servicio.

Los resultados obtenidos permiten concluir que el sistema desarrollado puede ser usado para la búsqueda de imágenes basadas en contenido, actualizando al mismo tiempo una Base de Datos que permita almacenar y recuperar información de imágenes digitales y URLs.

El artículo consta de cinco secciones incluyendo esta introducción. En la sección 2 se muestran los principales conceptos acerca de Recuperación de Imágenes Basada en Contenido, y acerca de las tecnologías actuales de Motores de Búsqueda. En la sección 3 se muestra el diseño del sistema, donde se exponen los componentes de la arquitectura del sistema y una descripción detallada de sus módulos.

En la sección 4 se muestra un ejemplo del funcionamiento del mismo mediante las interfaces de usuario, y el ambiente de desarrollo y pruebas. Finalmente en la sección 5 se presentan los resultados, conclusiones y recomendaciones para trabajos futuros.

2 Conceptos Fundamentales

En esta sección se muestran los conceptos fundamentales de los sistemas CBIR y de las tecnologías actuales de Motores de Búqueda de Imágenes para el Web.

2.1 Sistemas CBIR

Por sistemas CBIR se entiende una serie de tecnologías que permiten al usuario almacenar, indexar, clasificar y posteriormente recuperar información de imágenes digitales[7, 2]. El corazón de los sistemas CBIR es la obtención de imágenes utilizando la información presente dentro de las mismas imágenes, sin que sea estrictamente necesario recurrir a información textual acerca de las mismas ni a meta–información.

Debido a que los sistemas CBIR se encargan de varias actividades anteriores y posteriores a la recuperación de las imágenes, es común caracterizarlos como sistemas modulares donde cada uno de los módulos se encarga de una o varias primitivas de procesamiento, entre las que destacan:

1.- **Extraer características de las imágenes**: Lo que se hace de forma automática o semi–automática, de manera que las características extraídas sirvan de índices para la posterior clasificación y búsqueda. En [11, 19, 18] se presentan ejemplos de sistemas que emplean diferentes tipos de técnicas para la extracción de características de imágenes digitales. El resultado de la extracción de características de las imágenes se conoce como *vector característico* y es la representación numérica de las propiedades obtenidas por los algoritmos.

2.- **Almacenar los índices**: para esto se utilizan Estructuras de Datos y manejadores de Bases de Datos especializados para el manejo de información multidimensional, que es como usualmente se define la información proveniente de imágenes y otros objetos Multimedia. En general, de cada imagen se obtienen múltiples valores que representan en su conjunto una coordenada de un espacio N–dimensional. Muchas estructuras de datos se han definido para almacenar esta información multidimensional, tales como R–Trees, Q–Trees, SR–Trees, KD–Trees, etc [17].

3.- **Construir las solicitudes de búsqueda**: de acuerdo a un patrón o patrones proporcionados por el usuario o sistema cliente. Un compendio de técnicas se muestran en [3, 13]. Algunas propuestas manejan lenguajes de consulta textuales como MOQL [12]. Otras se basan en la construcción de prototipos de imágenes como base de la búsqueda como QBIC de IBM [1], mientras que otros se basan en la presentación de una o más imágenes de referencia, usando la técnica conocida como Búsqueda por Muestra (*Query by Example*) [11, 19, 18]. Una tendencia actual es la definición de protocolos para estandarizar las comunicaciones entre los clientes y los servidores. En [15, 14, 16] se presentan sistemas basados en el protocolo MRML, específicamente diseñado para CBIR.

4.- **Retornar los resultados del proceso de búsqueda**: Esta última fase se limita a la presentación de los resultados de las búsquedas efectuadas para el usuario. La forma, número de resultados, niveles de relevancia, información adicional que se presente a los clientes y otras consideraciones dependen de las especificaciones de las interfaces de usuario, y están en correspondencia directa con las capacidades de los algoritmos de búsqueda, las estructuras de almacenamiento y los lenguajes de especificación. Algunos sistemas otorgan especial importancia a la búsqueda por relevancia (*relevance feedback*) como puede verse en [10, 4].

Puede observarse que los diferentes pasos en la construcción y funcionamiento de los sistemas CBIR no indican nada acerca de la arquitectura del sistema. Se puede conceptualizar dichos sistemas como sistemas centralizados donde los diferentes pasos de funcionamiento los ejecuta un solo sistema modular, o como sistemas distribuidos en los que cada operación la realiza un elemento remoto. En general, en ambientes Cliente/Servidor, la especificación de la búsqueda se encuentra del lado del cliente, mientras que los algoritmos de búsqueda y las estructuras de almacenamiento se encuentran del lado del servidor.

2.2 Motores de Búsqueda

Los motores de búsqueda son programas computacionales cuya función es buscar constante y rápidamente páginas que se encuentren publicadas en Internet, y en general permiten realizar búsquedas por palabras o frases.

En los motores de búsqueda, la primera forma de organización de la información es mediante los llamados *directorios* que con intervención humana permiten organizar los sitios de búsqueda en función de temas de interés. En la Figura 1 se muestra un ejemplo de dicha organización en los buscadores Yahoo![2] y Google[3].

En general, los Motores de Búsqueda no ejecutan las solicitudes de los usuarios visitando el Web *en el mismo momento* que el usuario hace la consulta. Usualmente hacen una recolección sistemática de información *antes* que los usuarios hagan sus solicitudes.

La recolección se hace mediante un software llamado *crawler*, que se encarga de inspeccionar y clasificar la información de la red y almacenar los datos obtenidos de las páginas en una Base de Datos. Esta Base de

[1] http://wwwqbic.almaden.ibm.com
[2] http://www.yahoo.com
[3] http://www.google.com

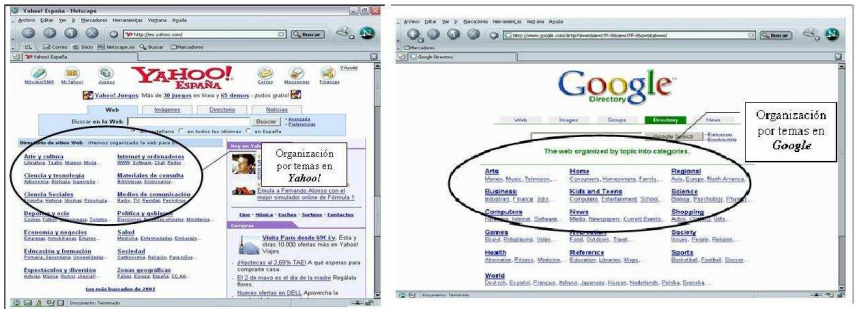


Figura 1: Organización de los directorios de Yahoo! y Google

Datos necesita ser actualizada constantemente, de lo contrario no estará sincronizada con las páginas reales y proveerá resultados obsoletos.

2.3 Motores de Búsqueda para Imágenes

Los Motores de Búsqueda de Imágenes, como su nombre lo indica, se utilizan para encontrar imágenes de cualquier tipo: fotos, dibujos, iconos, etc. Su localización implica un proceso diferente al utilizado para encontrar una página de texto, archivos de música, software, etc. Los motores de búsqueda de imágenes actuales se basan en la información que acompaña a la imagen, como puede ser el nombre del archivo en el que se encuentra la imagen, el texto que se encuentra a un lado de la imagen, que generalmente es el título que se quiere mostrar al usuario, etc.

Los buscadores de imágenes mantienen un catálogo con la información de las imágenes encontradas. Algunos permiten buscarlas en la forma como lo hace un directorio, otros por términos de búsqueda, o una combinación de ambos. Las imágenes que aparecen en los documentos HTML son fáciles de rastrear, debido a que existe una instrucción especial para usarlas.

Los Motores de Búsqueda más conocidos como Google, Yahoo!, etc., no hacen búsquedas de imágenes basadas en contenido, sino que se basan en la información textual que acompaña a la imagen. Como ejemplo de lo anterior en el Motor de Búsqueda de Imágenes de Google se colocó la palabra "clei" en el cuadro de texto como criterio de búsqueda. El resultado de la búsqueda de imágenes (Figura 2) que se aprecia no es del todo preciso, ya que las imágenes no se ajustan a los resultados esperados.

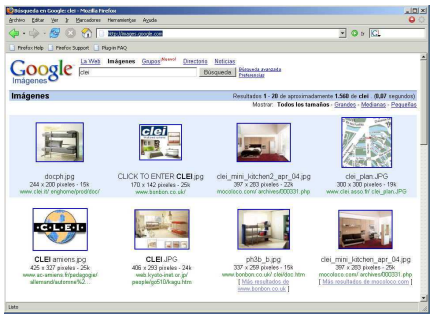


Figura 2: Resultado de Búsqueda de Imágenes en Google

A pesar de sus limitaciones, una ventaja de los buscadores actuales es que mantienen en sus Bases de Datos imágenes de una buena porción del Web, tanto como sus *crawlers* puedan recorrer, lo que resulta en un universo de búsqueda extenso y no limitado a Bases de Datos locales.

2.4 Motores de Búsqueda CBIR

La necesidad de buscar imágenes basándose en las características internas de las mismas está presente en muchos dominios comerciales y académicos, como la medicina, arquitectura, seguridad, arte y periodismo. En varias universidades y centros de investigación se han desarrollado sistemas que permiten buscar imágenes en base a las características de las mismas, generalmente características asociadas al color, textura y forma.

Algunos de los sistemas en desarrollo mejor documentados son:

2.4.1 PicSOM

PicSOM es parte de un proyecto para estudiar métodos y sistemas para la recuperación de imágenes basado en contenido en el Laboratorio de Computación y Ciencias de Información en la Universidad de Tecnología de Helsinski, Finlandia[9, 8].

Está basado en un algoritmo de red neuronal llamado Mapas Auto Organizativos (Self-Organising Maps, SOM). PicSOM usa una versión estructurada como un árbol del algoritmo SOM, para crear una representación jerárquica de la Base de Datos de las imágenes. Durante las consultas, los TS-SOMs (estructura de árbol de mapas auto organizativos) son usados para recuperar las imágenes similares de un conjunto dado de imágenes de referencia.

Como una base para recuperar las imágenes, el sistema PicSOM usa una combinación de varios tipos de rasgos estadísticos, que son procesados en la imagen origen, separando los rasgos en vectores que han sido formados para describir los colores, texturas y formas que se encuentran en las imágenes. Un demo está disponible en *http://www.cis.hut.fi/picsom/ftp.sunet.se*.

2.4.2 PicToSEEK

Es un sistema desarrollado por Theo Gevers y Arnold W.M. Smeulders, integrantes del Grupo de Investigación de Sistemas de Información Sensoriales Inteligentes (ISIS) de la Universidad de Ámsterdam, Holanda[5].

Para clasificar las imágenes PicToSeek básicamente toma en cuenta las siguientes características : Variación de color, Saturación del color y Transición fuerte del color.

En su primera etapa se recolectaron imágenes del web lo cual significa que PicToSeek posee un crawler autónomo. Estas imágenes se catalogaron automáticamente en varios y distintos tipos: por formato (JPEG-GIF), por escala de grises, profundidad de color, tamaños y fecha de creación. Luego, las imágenes se clasificaron según ciertos criterios para ser indexadas, dependiendo de las características de los colores.

Una vez que las imágenes han sido automáticamente recolectadas, catalogadas e indexadas, el sistema permite hacer búsquedas combinando: exploración visual a través de un catálogo, consulta por imagen ejemplo y consulta por características de la imagen. El proceso de recuperación de imágenes se hace de manera interactiva e iterativa, guiado según las características de las imágenes que ha elegido el usuario. Un demo del sistema está disponible en: *http://zomax.wins.uva.nl:5345/ret_user/*.

2.4.3 FIRE

Desarrollado por Thomas Deselaers, Daniel Keysers y Hermann Ney de la Universidad de Aachen, Alemania[1], utiliza una variedad de técnicas para la caracterización de imágenes, que incluyen el uso de características de color, textura y forma, además de incorporar *Relevance Feedback*.

Entre las técnicas cromáticas se encuentran el uso de histogramas de color y texturas, el uso de descriptores de textura basados en Matrices de Coocurrencia, y el uso de kernels de convolución para la extracción de contornos.

El sistema FIRE es Open Source, y puede ser adaptado a las necesidades de cada grupo de trabajo o usuario. Una versión en línea del sistema está disponible en: *http://www-i6.informatik.rwth-aachen.de/~deselaers/cgi_bin/fire*.

En el sistema FIRE, las imágenes son tomadas de una Base de Datos local que es un subconjunto de la Base de Datos de Corel (1000 imágenes).

En la Figura 3 se muestra un ejemplo de interacción de los sistemas PicSOM, PicToSeek y FIRE.

Los tres sistemas mostrados buscan imágenes de acuerdo al contenido, pero dentro de un universo de búsqueda limitado a sus Bases de Datos de muestra. Por otro lado, los buscadores como Google o Yahoo! tienen un universo de búsqueda muchos más amplio con respecto a imágenes, pero limita sus búsquedas al aspecto textual y de metainformación de las mismas. Un sistema que combine la búsqueda de imágenes por

Figura 3: Ejemplo de búsqueda en PicSOM, PicToSEEK y FIRE

contenidos con la búsqueda en el web puede ser beneficioso para muchas aplicaciones. En la siguiente sección se muestra el sistema HuyaWeb, que propone una solución a esto.

3 El sistema HuyaWeb

HuyaWeb es parte de un proyecto para la creación de sistemas de recuperación de imágenes basados en contenido, tanto en ambientes locales como en ambientes Web.

El sistema se basa en una arquitectura Cliente-Servidor, donde cada uno de sus componentes tiene tareas bien definidas. Los principales componentes de la arquitectura son:

- **Crawler:** Su función es recolectar imágenes en formatos jpg, jpeg, bmp, gif, png, xbm y xmp desde el Web, comenzando con una página Web como punto de inicio. Su funcionamiento se detalla en la sección 3.1. Para su desarrollo se empleó el lenguaje de programación Perl. Los tipos de página que indexa son: html, asp, jsp, shtml y php.
- **Aplicación Cliente:** Su función es facilitar a los usuarios la recuperación de imágenes mediante consultas al Servidor. Este módulo se desarrolló en JSP.
- **Aplicación Servidor:** El Servidor fue desarrollado usando tecnología Web JSP y el lenguaje de programación Java. La función de esta aplicación es procesar las solicitudes de imágenes de los clientes y proveer respuestas que satisfagan los requerimientos solicitados, mediante consultas a la Base de Datos o mediante visitas *on line* al Web.
- **Base de Datos:** Este módulo almacena eficientemente la información de imágenes de cada una de las páginas indexadas y recuperadas por el crawler. Para el desarrollo de la Base de Datos se usó el manejador de Base de Datos MySQL.

3.1 Módulo del Crawler

El sistema tiene implementado un *crawler* que busca, caracteriza e indexa imágenes de manera automática. En la Figura 4 se muestra el esquema general de funcionamiento del crawler. A continuación se describirán los pasos que el crawler realiza para indexar información de páginas Web.

1.- Se toma como entrada el URL de la página Web base. En el ejemplo de la Figura 4 la página donde se inicia el proceso de indexacción es: `http://www.spider.com/index.html` .

2.- El crawler comienza a descargar la página base y a partir de este documento comienza a buscar e indexar recursivamente todas las páginas Web e imágenes a las cuales ésta hace referencia (pasos 3 y 4). Para efectos del ejemplo de la Figura 4, los enlaces son `link1`, `link2`, y `link3` con sus respectivas imágenes: `imagen1.gif`, `imagen2.gif` e `imagen3.gif`, hasta llegar al límite definido por el usuario, que puede ser un número de imágenes descargadas o páginas Web visitadas.

El crawler se implementó usando rastreo en amplitud, de manera que el universo de búsqueda creciera de manera rápida hacia múltiples destinos del Web.

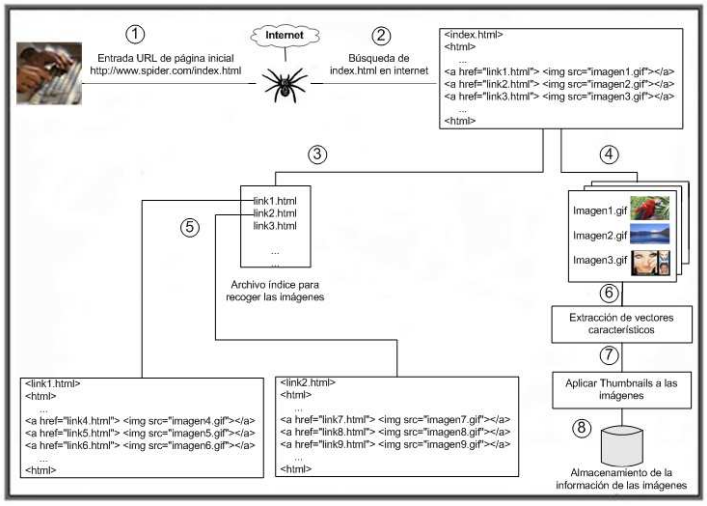


Figura 4: Esquema general de funcionamiento del crawler

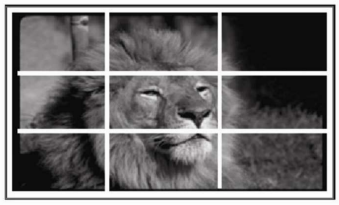


Figura 5: Proceso de división por nueve regiones

3.- En el paso 5 se comienza recursivamente el rastreo por amplitud de todos los enlaces que están en la página `link1.html` que son: `link4.html`, `link5.html` y `link6.html`. Una vez rastreados todos estos enlaces de la página `link1.html`, se procederá a rastrear por amplitud y recursivamente todos los enlaces que están en la página `link2.html`, para posteriormente seguir con los enlaces que están en la página `link3.html`.

4.- De cada imagen descargada se procede a extraer los vectores característicos (paso 6). En nuestro caso de prueba, el proceso de caracterización se realiza mediante el cálculo del promedio de las componentes RGB de la imagen. Para ello la imagen es dividida en nueve regiones de igual área y por cada región se obtienen tres valores (los promedios de las componentes *Red*, *Green* y *Blue*), por lo que el vector característico tiene veintisiete componentes enteros.

   En la Figura 5 se muestra un ejemplo de dicha división por regiones. La construcción modular del sistema permite incorporar cualquier algoritmo de extracción de características, por lo que el sistema puede ser adaptado fácilmente a necesidades específicas de búsqueda, como búsquedas basadas en texturas o en formas.

5.- Al momento de realizar el almacenamiento de las imágenes, en el paso 7, el crawler obtiene una imagen reducida (*thumbnail*) de cada una de las imágenes caracterizadas. La imagen reducida tiene un ancho o un alto máximo de 100 pixeles.

6.- El último paso consiste en almacenar la información de la imagen en la Base de Datos: el vector característico de la imagen original, la imagen en formato reducido, y los respectivos URL de la página y de la imagen.

Para evitar la repetición de imágenes en la Base de Datos se tomó el siguiente criterio: si existen dos (o más) imágenes con un mismo vector característico entonces se almacena en la Base de Datos únicamente

una de las imágenes, el URL de origen y el vector característico asociado a esa imagen. Para las imágenes consideradas como repetidas se almacena únicamente el URL donde se encuentre ubicada la imagen. De esta manera para un solo vector característico pueden existir varios URL de imágenes asociadas y las páginas donde se alojan.

3.2 Módulo del Cliente

Para realizar la búsqueda de imágenes, el usuario deberá proveer como entrada una imagen de consulta. El sistema ofrece dos opciones: la búsqueda `on line` y la búsqueda en la Base de Datos del servidor.

En la primera opción (on line) el usuario coloca además de la imagen una dirección Web sobre la cual el crawler iniciará el proceso de búsqueda de imágenes y extracción de características, ingresando además un número máximo de imágenes o de páginas Web a visitar. La búsqueda `on line` hace que el servidor active el crawler, de manera que los resultados son obtenidos directamente del web, lo que incide negativamente en el tiempo de respuesta pero al mismo tiempo garantiza una búsqueda actualizada y en los sitios específicamente dispuestos por el usuario. Las imágenes de respuesta se obtienen todas directamente del Web, es decir, el universo de búsqueda está restringido a aquellas páginas visitadas por el crawler en el momento de la solicitud. Las imágenes obtenidas además se almacenan en la Base de Datos local.

Como segunda opción se pueden realizar búsquedas de imágenes en la Base de Datos del servidor, ingresando una imagen y opcionalmente un URL. La Base de Datos es generada previamente por el proceso de búsqueda del crawler y está disponible para el usuario, de manera análoga a como lo hacen los motores de búsqueda usuales. Aunque esto hace más eficiente el proceso de búsqueda, los resultados podrían estar desactualizados, pero aumenta considerablemente el universo de búsqueda. El URL opcional permite delimitar las respuestas a aquellas imágenes que estén alojadas en determinados sitios.

Las imágenes resultado se presentan en forma de *thumbnails*, ordenadas de acuerdo a la similitud con respecto a la imagen base provista por el usuario. Este módulo se desarrolló con tecnología JSP.

3.3 Módulo del Servidor

Este módulo se encarga de activar el crawler para la búsqueda e indexación de páginas Web, extrae las características y almacena las imágenes en la Base de Datos.

Al momento de indexar la información referente a las imágenes, se almacena en la Base de Datos la imagen, su vector característico, el URL de imagen y URL de página Web donde está alojada la imagen. La ventaja de almacenar toda esta información es que al actualizar la Base de Datos con la inserción de un nuevo algoritmo para la extracción de características, no será necesario que el crawler recorra nuevamente todas las páginas Web, porque ya estará almacenada toda la información en la Base de Datos local. Como resultado se obtendrán respuestas de consultas de imágenes más rápidas a los usuarios, pero no se ahorran recursos de almacenamiento, ya que se almacenan todas las imágenes, sus vectores característicos, el URL de imagen y URL de página Web.

El servidor recibe las consultas de los usuarios en forma de una imagen de muestra. Esta imagen es caracterizada y comparada contra las imágenes presentes dentro de la Base de Datos del Servidor o contra el universo de búsqueda obtenido en línea. La comparación se realiza obteniendo la distancia euclideana entre los vectores característicos de la imagen de prueba y las imágenes del universo de búsqueda. A menor la distancia, más similares se consideran las imágenes.

El servidor se implementó usando Java con llamadas a módulos Perl y fue alojado usando el servidor HTTP Resin.

3.4 Base de Datos

La Base de Datos se desarrolló en MySQL y está conformada por tres tablas que se van llenando cuando el módulo del crawler está indexando páginas Web, para su posterior recuperación por el módulo del servidor. En la Figura 6 se establecen las relaciones entre las tablas del sistema.

Como puede observarse en la tabla `data_image` una imagen puede tener muchos vectores característicos en la tabla `data_vector`, estableciéndose una relación de uno a muchos.

La tabla `data_vector` tiene un campo `link_image` que guarda los enlaces de imágenes indexadas por el crawler. Este campo tiene una relacion de uno a muchos con el campo `link_image` de la tabla `data_crawler`.

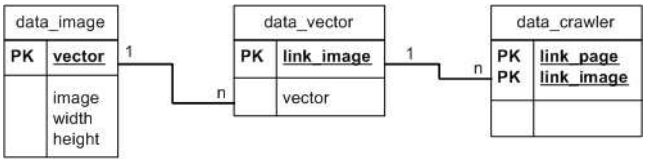


Figura 6: Relaciones entre las tablas de la Base de Datos

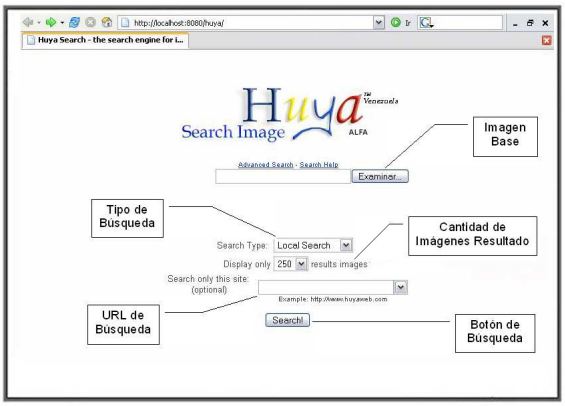


Figura 7: Interfaz Inicial

La tabla `data_crawler` almacena enlaces de imágenes y URLs como claves primarias. No se permite en esta tabla que existan dos tuplas de registros iguales en la Base de Datos del sistema.

4 Ejemplo de uso del sistema HuyaWeb

A continuación se presentan las interfaces gráficas para las búsquedas de imágenes basadas en contenido.

En la Figura 7 se muestra la pantalla inicial para realizar las búsquedas por parte de los clientes. En esta pantalla se solicitan los parámetros necesarios para hacer las consultas al servidor HuyaWeb.

La interfaz inicial permite realizar dos tipos de consulta: consultas sobre la Base de Datos y consultas On Line. Por defecto la opción habilitada es la búsqueda en la Base de Datos.

4.1 Consulta en la Base de Datos

Los parámetros para la consulta en la opción de Base de Datos son: la imagen de consulta y opcionalmente un URL. En la Figura 8(a) se muestra la interfaz para este tipo de búsqueda.

El parámetro obligatorio es la imagen que servirá como base para comparar similitud. En caso de indicarse el parámetro de URL, la consulta en la Base de Datos se realiza únicamente sobre las imágenes pertenecientes a páginas con dicho URL como base, en caso contrario se busca en toda la Base de Datos. Por ejemplo, si el usuario especifica el URL “www.misitio.com” la búsqueda se realiza únicamente sobre las imágenes cuyo URL comience con “www.misitio.com”.

Una vez seleccionados los parámetros, el cliente efectúa la petición al servidor a través del botón ”Search”.

Como se ha mencionado, el parámetro URL permite limitar los resultados de la búsqueda a un dominio particular dentro de la Base de Datos. Es importante destacar que si se especifica ese parámetro y dicho dominio no ha sido indexado previamente, entonces no se arrojan resultados de similitud de imágenes.

4.2 Consulta On Line

Las funcionalidades de interfaz de las búsquedas son básicamente iguales para las consultas de Base de Datos y para la búsqueda On Line.

En la Figura 8(b) se muestra la pantalla para realizar búsquedas en la opción On Line. Los parámetros para la consulta son: imagen, URL y número de páginas o número de imágenes a indexar. Se requieren como parámetros obligatorios la imagen, el URL base y al menos un criterio de parada que proporciona el usuario, que puede ser un máximo de imágenes o un máximo de páginas. Debe recordarse que este criterio de parada es el que le indicará al crawler cuándo detenerse durante el proceso recursivo de búsqueda y descarga de imágenes desde el Web.

Una vez seleccionados estos parámetros, el cliente efectúa la petición al servidor a través del botón "Search".

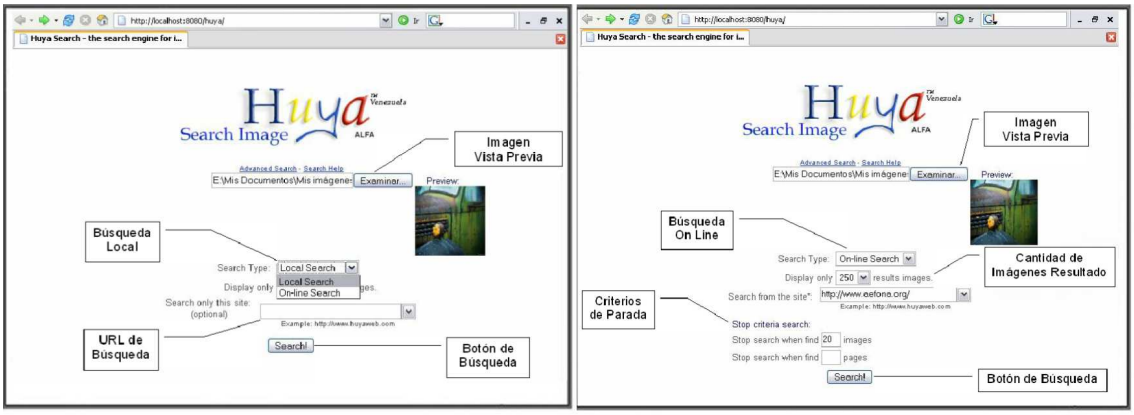


Figura 8: (a) Consulta en Base de Datos y (b) Consulta On Line.

4.3 Resultados de las Consultas

Las interfaces de resultados de las búsquedas son iguales para las consultas de Base de Datos y para la On Line. En la Figura 5 se muestra la interfaz que presenta un conjunto de imágenes similares a una imagen dada por el cliente.

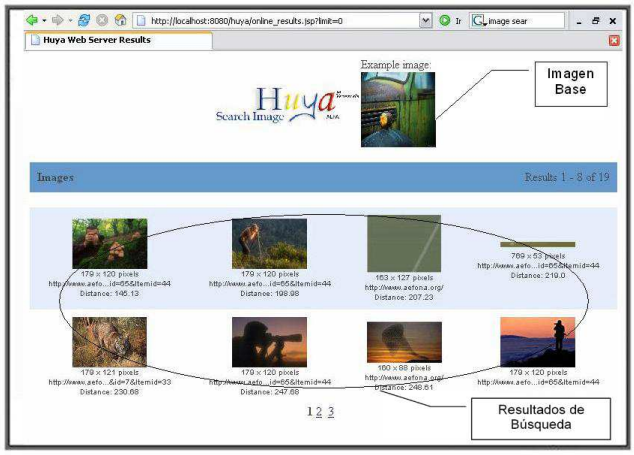


Figura 9: Resultados de una consulta.

Esta interfaz presenta las imágenes en tamaño reducido y sobre cada imagen se puede hacer click para ir a la página Web de donde originalmente se descargó dicha imagen.

Por cada imagen se muestra lo siguiente: dos números que indican el tamaño en ancho y alto en pixeles, el URL origen donde se encuentra la imagen en el Web y un número que representa la distancia euclideana entre dicha imagen y la imagen de muestra original.

La validez de los resultados depende del algoritmo específico de extracción de características y de la función de distancia para el cómputo de la similitud.

5 Resultados y Trabajos Futuros

En el estado actual del proyecto, los principales aportes de nuestro trabajo son:

- El sistema desarrollado representa un aporte en cuanto a sistemas de búsqueda de imágenes basados en contenido, ya que permite que los usuarios busquen imágenes similares a las de muestra sobre cualquier sitio Web disponible públicamente y no únicamente sobre Bases de Datos locales a los servidores.
- La construcción modular del sistema permite incorporar nuevas técnicas de caracterización y estructuras de datos de almacenamiento. Se hicieron pruebas con diferentes algoritmos de extracción de características que muestran que el sistema puede ser adaptado para requerimientos de usuarios en campos específicos, como búsquedas por forma o por textura.
- HuyaWeb permite almacenar en Bases de Datos locales al servidor las imágenes que se descargan del Web. Esto facilita el proceso de construcción de repositorios de imágenes para su posterior ofrecimiento a los usuarios, tal como hacen los sitios de búsqueda actuales con las páginas Web.
- La opción de búsqueda `on line` permite que los usuarios hagan sus consultas sobre sitios específicos del web e incluso sobre sitios de persistencia efímera, al costo de un tiempo mayor para la caracterización de las imágenes. Esta alternativa no ha sido reportada antes en otros sistemas CBIR en línea.

5.1 Limitaciones del Sistema

El prototipo desarrollado presenta las siguientes limitaciones:

- El crawler está desarrollado en Perl mientras que el resto de la programación se desarrolló en Java, por lo cual no hay suficiente control entre las llamadas de procesos.
- HuyaWeb no realiza la actualización periódica de la información de la Base de Datos contra la información del Web, por lo que puede mostrar resultados obsoletos.
- No se hace búsqueda por relevancia, es decir no se pueden hacer nuevas búsquedas de imágenes basado en contenido en función de resultados de búsquedas previas.

5.2 Trabajos Futuros

Actualmente se trabaja en los siguientes aspectos:

- Desarrollar el módulo del crawler en Java para permitir un mayor control y eficiencia entre el servidor y el crawler.
- Realizar estudios de usabilidad y satisfacción de usuarios.
- Incorporar más algoritmos de extracción de características y mejores técnicas de interacción con el usuario, por ejemplo *relevance feedback*.

Agradecimientos

Partes del código fueron desarrolladas por estudiantes del Centro de Computación Paralela y Distribuida de la UCV, en especial se agradece la colaboración de Yeny Hernández, Marlyn Castro y César González. El crawler en Perl es una adaptación de un trabajo original de Lee Goddard [6].

Referencias

[1] T. Deselaers, D. Keysers, and H. Ney. Fire - flexible image retrieval engine: Imageclef 2004 evaluation. In *Proceedings of the CLEF 2004 Workshop. Bath, UK*, pages 535–544, September 2004.

[2] J. Eakins and M. Graham. Content–based image retrieval. *Technical Report JTAP–39, University of Northumbria at Newcastle*, October 1999.

[3] M. Fernández, J. Simeon, P. Wadler, S. Cluet, A. Deutsch, D. Florescu D., A. Levy, D. Maier, J. McHugh, J. Robie, D. Suciu, and J. Widom. XML query languages: Experiences and exemplars. communication to the w3c., September 1999.

[4] J. French, W. Martin, and J. Watson. A qualitative examination of content–based image retrieval behavior using systematically modified test images. In *45th IEEE International Midwest Symposium on Circuits and Systems, Tulsa*, August 2002.

[5] Theo Gevers and Arnold Smeulders. PicToSeek: Combining color and shape invariant features for image retrieval. *IEEE Transactions on Image Processing*, 9(1):102–119, January 2000.

[6] Lee Goddard. Cpan. http://search.cpan.org/l̃goddard/HTTP-GetImages-0.343/GetImages.pm, 2004.

[7] A. Goodrum. Image information retrieval: An overview of current research. *Journal of Informing Science – Special Issue on Information Science Research*, 3(2), 2000.

[8] PicSOM Development Group. Picsom's user's guide. *Helsinski University of Technology, http://www.cis.hut.fi/picsom/picsom.html*, june 2001.

[9] Markus Koskela, Jorma Laaksonen, Sami Laakso, and Erkki Oja. The PicSOM retrieval system: Description and evaluations. In *J. P. Eakins and P. G. B. Enser, eds., Challenge of Image Retrieval 2000, Brighton, UK*, May 2000.

[10] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja. Self–organizing maps as a relevance feedback technique in content–based image retrieval. *Pattern Analysis and Applications*, 4(2):140–152, June 2001.

[11] Jorma Laaksonen, Markus Koskela, Sami Laakso, and Erkki Oja. PicSOM – content-based image retrieval with self–organizing maps. *Pattern Recognition Letters*, 21(13-14):1199–1207, 2000.

[12] J. Z. Li, M. T.Özsu, D. Szafron, and Vincent Oria. MOQL a multimedia object query language. *Technical Report TR–97–01, Department of Computing Science, University of Alberta, Canada*, January 1997.

[13] Peiya Liu and Liang H. Hsu. An approach to specifiyng and querying multimedia objects and scheduled structures in XML documents. In *Spatial/Temporal Databases Meeting, Paris.*, 2000.

[14] Henning Müller, Antoine Geissbuhler, and Stéphane Marchand-Maillet. Extensions to the multimedia retrieval markup language –a communication protocol for content–based image retrieval. In *Proceedings of the CBMI*, September 2003.

[15] Wolfgang Müller, Henning Müller, and Stéphane Marchand-Maillet. MRML: A communication protocol for content-based image retrieval. In *International Conference on Visual Information Systems. Lyon, France*, November 2000.

[16] Rivas-Suárez, Yeny Hernández, and Marlyn Castro. HUYA : Un sistema para recuperación de imágenes basado en MRML. *Memorias de la XXX Cconferencia Latinoamericana de Informática*, pages 777–789, September 2004.

[17] Y. Rui, T. S. Huang, and S. F. Chang. Image Retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, 1999.

[18] John R. Smith and Shih-Fu Chang. VisualSEEK: A fully automated content-based image query system. In *ACM Multimedia*, pages 87–98, 1996.

[19] J. Z.Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics–sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(9):947–963, 2001.

Servidor Genérico Para Portales De Voz (SGVP)

Diana Maritza Muñoz
Universidad de Los Andes, Dept. Ingeniería de Sistemas y Computación,
Bogotá, Colombia
dian-mun@uniandes.edu.co

and

Harold Cruz Ortiz
Universidad de Los Andes, Dept. Ingeniería de Sistemas y Computación,
Bogotá, Colombia
hacruz@uniandes.edu.co

Abstract

This paper shows the details of design and implementation of voice gateway VoiceXML based. The implemented system has an architecture in which native developments with the reusability of components of software GNU are combined which they could be integrated to the system like result of an exhaustive process of reverse engineering. The conducted tests allowed to verify the operation of the system and the fulfillment of the proposed objectives.

Keywords: Voice portals, VoiceXML, SIP, Voice portal servers, convergence.

Resumen

Este artículo presenta los detalles de diseño e implementación de un gateway de voz basado en VoiceXML. El sistema implementado tiene una arquitectura en la que se combinan desarrollos nativos con la reutilización de componentes de software GNU que pudieron ser integrados al sistema como resultado de un proceso de reverso ingeniería exhaustivo . Las pruebas efectuadas permitieron verificar el funcionamiento del sistema y el cumplimiento de los objetivos propuestos.

Palabras claves: Portales de voz, VoiceXML, SIP, Servidores para portales de voz, Convergencia

1. Introducción

La aparición de Internet, ocasionó un cambio tecnológico y de concepción a nivel mundial en temas como la manipulación, disponibilidad y flexibilidad de la información, aunque en sus inicios este mundo de los datos se creyó siempre aislado e independiente del de la voz. Durante los siguientes años, la necesidad de prestar mejores servicios a los usuarios, impulsó a los sectores de informática y comunicaciones a buscar la convergencia entre sus diferentes tecnologías de red, lo que originó un segmento conocido como integración de informática y telefonía o CTI [4][7][8][44]. Las aplicaciones CTI fueron la primera aproximación entre las redes de voz y datos, ofreciendo servicios de valor agregado que facilitaron la interacción con los usuarios. Uno de los problemas fundamentales en este tipo de aplicaciones es que están basados en arquitecturas propietarias de los fabricantes, lo que dificulta la definición de estándares y por consiguiente el funcionamiento en diferentes plataformas. El siguiente paso fue la transmisión e integración de voz y datos sobre las mismas redes. La tecnología de VoIP [33][47] permitió que un medio que exclusivamente se había pensado para transporte de datos, se encargara también de llevar paquetes de voz demostrando que esa barrera antes impuesta, carecía de sentido y que las posibilidades que ofrece a los usuarios la interacción de la voz y los datos son muy amplias.

El objetivo siguiente fue hacer estos dos mundos transparentes para el usuario de modo tal que la biblioteca de información mundial, pueda ser accedida desde cualquier punto y mediante el empleo de cualquier mecanismo de conexión. Razón por la cual, los miembros del W3C (World Wide Web Consortium) decidieron centrar su atención en la búsqueda de nuevos servicios que permitan un cambio en la filosofía de interacción con el usuario para que sean los sistemas de comunicación, los que se adapten a sus necesidades, surgiendo entre las opciones analizadas la incorporación de la voz como medio de acceso a la Web [3], lo que conllevaría a la ampliación de la población objetivo y a la posibilidad de prestar una mejor atención a los usuarios finales.

Esta incorporación se pudo llevar a cabo en los portales de voz [40] por el avance que han tenido últimamente las técnicas de reconocimiento de voz [24][25] y las de síntesis de voz [12][24], que permiten a los usuarios comunicarse de manera natural con un sistema, el cual identifica lo que se le está solicitando, procesa la solicitud y responde al usuario por el mismo medio haciendo transparente el acceso a los datos. Esta es la razón fundamental por la cual se realiza la presente investigación, en la que se pretende ofrecer un sistema genérico con una arquitectura de fácil escalamiento y adaptación, basada en estándares, modular y que permita independizar la capa de servicios de la capa de red, que le brinde a la comunidad otra alternativa de acceso en la cual mucha más gente pueda interactuar con la información que se encuentra disponible en Internet.

2. Descripción del Sistema

El Servidor Genérico para Portales de Voz o SGPV es un sistema que permite aprovechar las facilidades y servicios prestados por dos de los principales sectores de las telecomunicaciones en el mundo, la telefonía e Internet, ver figura 1.

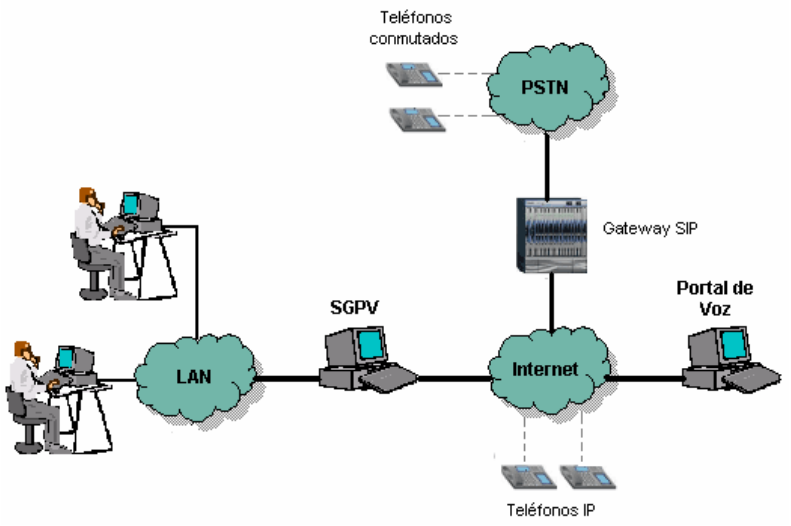


Figura 1. Contexto de funcionamiento del SGPV

Las funcionalidades que ofrece el sistema son:

- Permite prestar los servicios que se encuentran disponibles en Internet a usuarios de telefonía fija y móvil, al igual que a los usuarios conectados a través de un PC por medio de simuladores de telefonía que implementen alguno de los estándares soportados por el sistema (SIP y CAPI).
- Posibilidad de navegación del portal a través de comandos de voz o tonos multifrecuenciales del teléfono (DTMF).
- Navegación por Internet o local de contenidos desarrollados empleando el lenguaje VoiceXML.
- Captura de eventos específicos para el manejo de comandos especiales disponibles para los usuarios como la ayuda o salir del sistema.

De las características de la implementación desarrollada, es conveniente anotar las siguientes:

- El sistema permite el acceso concurrente de múltiples usuarios al portal, distribuyendo los recursos entre los abonados.
- Es escalable, debido a su arquitectura modular, que permite que nuevos módulos sean adicionados al sistema fácilmente. Asimismo, es posible cambiar los módulos existentes y reemplazarlos sin mayores ajustes al sistema.
- Soporta diferentes servidores de telefonía que pueden operar al mismo tiempo, de manera que los usuarios puedan conectarse mediante la interface ISDN que posee el sistema y por medio del protocolo SIP simultáneamente.
- La configuración de qué motor de reconocimiento de voz, de síntesis de voz y que plataforma de telefonía se va a utilizar se puede establecer en el archivo de configuración del sistema, para que en tiempo de ejecución se carguen los recursos necesarios para su funcionamiento.
- Es posible adicionar varios módulos de reconocimiento de voz, mediante la implementación del API del sistema.
- Es posible adicionar varios módulos de síntesis de voz, mediante la implementación del API del sistema.
- Es posible adicionar varios servidores de telefonía mediante la implementación del API de hardware de telefonía del sistema.

3. Análisis de los requisitos

Antes de plantear la arquitectura del SGPV, se detallaron las características de diseño que deben tenerse en cuenta para la construcción de los portales de voz, con base en estas características se definieron los requerimientos del SGPV que fueron la base para el diseño de la arquitectura.

3.1. Diseño de un portal de voz

La parte fundamental en el proceso de diseño y desarrollo de una aplicación Web con navegación por voz, es visualizar la aplicación como un diálogo entre el usuario y el sistema [9][32]. Es necesario identificar cómo va a interactuar el usuario con el sistema, si lo hará mediante comandos de voz, en lenguaje natural o a través de la manipulación del teclado telefónico. Posteriormente, se debe identificar qué es lo que la aplicación le va a reproducir al usuario (los diálogos) y la forma en la que lo va a hacer, si es a través de archivos pregrabados o con la salida provista por el motor de síntesis de voz o TTS. Otro aspecto importante es qué respuestas puede proporcionar el usuario al sistema y cómo se va a manejar la información que él proporciona visualizando en qué partes de la aplicación el usuario puede ingresar información al sistema, si tendrá la opción de emitir comandos especiales durante su sesión o si se desea soportar sinónimos para los comandos o soportar lenguaje natural.

Una vez se han establecido las palabras o frases con las que el usuario puede responderle al sistema, es necesario definir la gramática para esas palabras. Una buena opción para la consolidación definitiva del sistema, es experimentar con diferentes vocabularios para validar el diseño que mejor satisfaga los requerimientos y expectativas de los usuarios

Por último es necesario determinar qué va a hacer el sistema con el texto proporcionado por el módulo de reconocimiento de voz que contiene la solicitud hecha por el usuario. En este punto, se deben tener en cuenta los posibles errores que se pueden presentar durante la fase de reconocimiento, por ejemplo cuando el usuario dice algo que no está en la gramática establecida, cuando la voz del usuario es inteligible o cuando el usuario no le responde al sistema.

Teniendo en cuenta estas características, se determinaron los requerimientos del SGVP.

3.2. Requerimientos del SGPV

Los requerimientos básicos de un servidor de portales de voz son los siguientes:

Recibir llamada: permitir que los usuarios puedan comunicarse al número telefónico destinado para la ejecución de la aplicación.

Establecer llamada: una vez llegue una llamada al sistema, debe establecerse el canal de comunicación entre el abonado y el sistema que será utilizado para el transporte de la información.

Ejecutar Script de Diálogo: cuando se haya establecido el canal de comunicación, el sistema debe cargar el archivo que contiene la descripción del diálogo con base en el archivo de configuración del sistema e iniciar su ejecución. Esta ejecución puede implicar la reproducción de archivos de audio, recolección de audio o de tonos multifrecuenciales del teléfono, grabación de voz, síntesis de voz, etc.

Reproducir Archivo de Audio: el sistema debe cargar el archivo de audio y repoducirlo en el canal existente entre el abonado y el sistema.

Reconocer DTMF: el sistema debe poder identificar los tonos multifrecuenciales enviados por el usuario.

Grabar Voz: el sistema debe poder obtener el audio emitido por el abonado en el canal de comunicación y almacenarlo en un archivo físico en el servidor.

Reconocer comandos de voz: el sistema debe realizar un proceso de reconocimiento de los comandos de voz emitidos por el usuario en el canal de comunicación.

Sintetizar Cadena de Texto: el sistema debe generar un archivo que contega el audio que representa una cadena de texto ingresada como parámetro.

Terminar llamada: El abonado que se comunica con el sistema puede terminar su llamada en cualquier momento. El sistema debe detectar que el canal de comunicación ha sido interrumpido y liberar los recursos utilizados por esa llamada. Además, el sistema también debe terminar la llamada una vez finalice la ejecución de la aplicación VoiceXML.

Los requerimientos no funcionales incluyen manejar la concurrencia para soportar varias llamadas al mismo tiempo, minimizar el tiempo de respuesta para poder obtener una comunicación en línea con el abonado y adicionalmente se desea tener en cuenta que se pueda extender la funcionalidad fácilmente y que contenga módulos estándar que le brinden flexibilidad a la aplicación.

A partir de los requerimientos mencionados anteriormente se estableció el diseño del SGVP.

3.3. Diseño del Sistema

Después de analizar el estado actual de los componentes, las investigaciones que hay respecto a los servidores para portales de voz y de tener en cuenta que el principal objetivo de esta investigación es desarrollar un servidor genérico para portales de voz, se tomaron las siguientes decisiones:

- De los protocolos de telefonía analizados: H.323 [13], SIP (Session Initiation Protocol) [10][28] y MGCP (Media Gateway Control Protocol) [29], se escogió SIP como protocolo para el establecimiento de sesión por las ventajas que ofrece como la flexibilidad en el tipo de información que se puede enviar una vez establecida la sesión, la poca sobrecarga en el proceso de establecimiento de la llamada y su amplio uso [42][43]. En la tabla 1 se muestra una comparación entre estos distintos protocolos.

	SIP	H.323	MGCP/Megaco
Filosofía	Horizontal	Vertical	Vertical
Complejidad	Baja	Alta	Alta
Alcance	Simple	Completa	Parcial
Escalabilidad	Buena	Pobre	Moderada
Posibilidad de Nuevos Servicios	Si	No	No
Enlace con Internet	Si	No	No
Compatibilidad con SS7	Pobre	Pobre	Buena
Costo	Bajo	Alto	Moderado

Tabla 1. Comparación entre SIP, H.323 y MGCP/Megaco. (Tomado de [35]).

  La implementación de este protocolo se hizo tomando como base el código libre desarrollado por la comunidad investigativa de Vovida[14][22][27].

- Para el transporte de la información se escogió RTP (Real-Time Transport Protocol) por varias razones, entre ellas tenemos el hecho de ser el protocolo de transporte más comúnmente usado en las aplicaciones de telefonía, el estar basado en IP, el proveer soporte para la transmisión de datos en tiempo real y los servicios de reconstrucción de tiempos, detección de pérdidas, seguridad e identificación de contenido que ofrece [30][31]. La implementación de este protocolo se hizo tomando como base el código libre desarrollado por la comunidad investigativa de Vovida [14][22].

- De los gestores de diálogo analizados: VoiceXML (Voice eXtensible Markup Language) [16][32] y SALT (Speech Application Language Tags) [20][21], se seleccionó fue VoiceXML debido a su amplio uso, a la diferenciación que permite hacer entre los datos, la interfaz con el usuario y la lógica de la aplicación [23] y porque a criterio personal es más que un lenguaje, es casi un protocolo porque define de manera estándar cómo interactuar con el usuario, cómo realizar el proceso de recolección y entrega de la información, sin definir el lenguaje de programación en el que debe hacerse, con la misma filosofía de HTML (HiperText Markup Language), por consiguiente permite además, que los actuales desarrolladores de aplicaciones Web puedan ingresar a esta nueva tecnología de manera casi

transparente y que sus desarrollos los puedan hacer en los lenguajes de programación preferidos por cada uno, teniendo como base VoiceXML.

- De las arquitecturas actuales de servidores de portales de voz estudiadas: Elvira [19], OpenVXI [1] y PublicVoiceXML [2], se escogió este último en su versión 3.0 como la arquitectura base para desarrollar la solución debido a su trayectoria, a su continuo avance y al soporte brindado por sus desarrolladores para la modificación o adición de funcionalidades. PublicVoiceXML es una aplicación multiplataforma (soporta diferentes versiones de Linux, Unix y Windows), de interacción por medio de voz que integra los servicios de voz con los datos y libera a los desarrolladores de los portales de voz de la programación de bajo nivel ya que provee una infraestructura que incluye la funcionalidad de interacción con el hardware de telefonía utilizando CAPI 2.0 (Common-ISDN-API) protocolo de comunicación definido para las líneas ISDN y con el motor de síntesis de voz (Festival [26], por ejemplo). PublicVoiceXML implementa casi la totalidad de la funcionalidad de VoiceXML 2.0 descrita en [18], logrando unir los scripts VoiceXML con las líneas telefónicas.

Otros aspectos considerados en el diseño del SGPV fueron el manejo de la concurrencia para poder manejar llamadas simultáneas que compartan los recursos del sistema y el tiempo de respuesta a las solicitudes hechas por el usuario que debe ser mínimo para que la interacción se haga en línea [34][39].

Con base en los requerimientos antes mencionados y en los aspectos de diseño definidos en esta sección se planteó la arquitectura del SGPV.

3.4. Arquitectura del Sistema

La selección de la arquitectura del SGPV se hizo después de un análisis de las estructuras propuestas en la actualidad, tomando como base la propuesta por los diseñadores de PublicVoiceXML [2]. En la Figura 2, se detalla el sistema propuesto para el Servidor Genérico para Portales de Voz, que busca satisfacer los requerimientos de las redes convergentes. A continuación se hará una breve descripción de cada uno de los módulos que hacen parte de esta arquitectura.

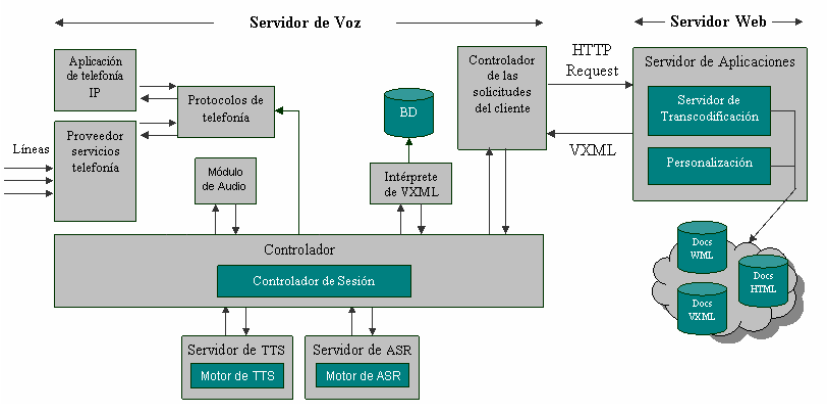


Figura 2. Arquitectura del Servidor Genérico Para Portales de Voz (SGPV)

3.4.1. Servidor de Voz

El Controlador es el coordinador del sistema en su totalidad y es el responsable de interactuar con el servidor Web y con el abonado. Además, sirve como interfaz entre diferentes módulos del servidor de voz y es el encargado de manejar el acceso concurrente al sistema. El manejo de la concurrencia se trabaja creando un hilo por cada una de las llamadas y creando para cada uno de estos hilos un intérprete de VoiceXML propio. Esta decisión se tomó para evitar el acceso a los mismos recursos en tiempo de

ejecución y para distribuirlos óptimamente, de manera que el Abonado B no deba esperar a que el intérprete de VoiceXML utilizado en la llamada del abonado A termine de ejecutar el script para poder ser atendida. PublicVoiceXML contiene una definición del controlador pero fue necesario modificarlo para poder extenderlo de acuerdo con la arquitectura propuesta. Se incluyó la interacción con el motor de reconocimiento de voz y con el módulo de protocolos de telefonía.

El Controlador de sesión maneja las llamadas simultáneas recibidas por el sistema y mantiene un registro del estado actual de cada una de éstas, hasta que la llamada finalice. Este controlador se acondicionó para que soporte el protocolo estándar de establecimiento de sesión SIP. Para poder realizar esta integración se empleó la estructura del patrón Proxy de manera que la implementación del protocolo SIP sea independiente del SGPV. El desarrollo del protocolo SIP implicó identificar cómo debe responder el sistema a cada uno de los mensajes SIP que le pueden llegar al sistema, manejar los estados posibles del sistema, crear los paquetes SIP con la información requerida por el sistema para establecer la sesión con el usuario e implementar esta funcionalidad para acoplarla al SGPV. Para la creación, envío y recepción de mensajes SIP y RTP se utilizaron los componentes libres de Vovida.

El Controlador de las Solicitudes del Cliente es usado por el Controlador para enviar solicitudes utilizando el protocolo HTTP al componente Web y recibir de éste los documentos VoiceXML. A este controlador no se le realizaron modificaciones, se utilizó el desarrollado por PublicVoiceXML.

El Intérprete de VoiceXML es quizá la parte esencial de los portales de voz, ya que es el encargado de darle significado a las etiquetas del lenguaje VoiceXML. El SGPV esta basado en el desarrollo de PublicVoiceXML lo cual garantiza que no haya extensiones propietarias al lenguaje, de manera que las aplicaciones desarrolladas sobre esta base cumplan con la especificación VoiceXML 2.0 del W3C .

El Módulo de Audio usa una API de sonido para reproducir audio en las líneas, de manera que diferentes tipos de archivos de audio pueden ser reproducidos. El sistema emplea archivos wav en formato PCM de 8 Kb/s mono, formato común para aplicaciones de telefonía.

El Proveedor de Servicios de Telefonía o TSP por sus siglas en inglés (Telephony Service Provider) es una aplicación provista por el fabricante del dispositivo de comunicación que se puede emplear, para el establecimiento, mantenimiento y desconexión de la llamada. Este módulo aplica en aquellos casos que se desee tener un servidor de telefonía conectado con una tarjeta telefónica específica.

El Módulo de Protocolos de Telefonía inicializa todas las líneas conectadas al computador donde se ejecuta la aplicación en caso de tener una arquitectura basada en un hardware específico de telefonía empleando el TSP dado por el proveedor, controla la comunicación y la conexión con las líneas. En este módulo se incluyó la implementación de SIP de manera que en este momento el SGPV soporta la comunicación mediante CAPI y SIP cubriendo una amplia variedad de opciones para interactuar con distintos dispositivos.

3.4.2. Servidor de Síntesis de Voz o TTS

El motor de síntesis de voz se encuentra ubicado en la arquitectura en un servidor independiente, por el alto consumo de recursos realizado durante su proceso. Cualquier motor de síntesis de voz puede ser enlazado al sistema mediante la implementación de la interface de síntesis de voz para que el texto pueda ser convertido a un archivo de sonido. El archivo obtenido por este módulo es entregado al modulo de Audio para ser reproducido por la línea correspondiente.

A este módulo se le adicionó la funcionalidad de convertir los archivos generados en un formato soportado por el protocolo de trasnporte, en este caso μ-law para que los paquetes puedan ser transmitidos al usuario. Para la conversión de formatos se utilizó la librería *libsndfile* [37], cuyo fin es leer y escribir ficheros de audio en diferentes formatos, a través de una interfaz común y estándar. Además, es capaz de leer y/o

escribir también ficheros de audio en formato PCM e incluso ficheros WAV, así como una amplia serie de formatos de audio comprimidos.

3.4.3. Servidor de Reconocimiento de Voz o ASR

Debido a la complejidad de los sistemas ASR y al alto costo en recursos requerido para su procesamiento, se aconseja que el motor de reconocimiento de voz se encuentre en un servidor independiente aunque podría incluirse dentro del mismo servidor dispuesto para el motor de síntesis de voz.

Actualmente el desarrollo de PublicVoiceXML no tiene desarrollada la interacción con los módulos de reconocimiento. La integración hecha en el SGPV con un motor de reconocimiento de voz se diseñó mediante una interface que define la funcionalidad esperada, incluyendo un método para la creación de la gramática requerida por el motor para efectuar la labor de reconocimiento. Además, se desarrolló la implementación de esta interface para que interactúe con el motor ASR de Microsoft Speech SDK 5.1, incluyendo la creación de la gramática en el formato SAPI requerida por este motor.

Cualquier motor de reconocimiento de voz puede ser enlazado al sistema mediante la implementación de la interface. La funcionalidad que debe cumplir es la siguiente: permitir reconocer el mensaje contenido en un archivo de audio pregrabado y entregar al Controlador el texto reconocido, para que éste lleve a cabo las acciones requeridas de acuerdo con la información entregada. De las características presentes en este módulo tales como la técnica empleada para el reconocimiento, la cantidad de vocabulario con el que cuenta, los mecanismos de eliminación de ruido de fondo, el requerir o no de un entrenamiento previo, entre otros, depende la manera en que el portal de voz va a interactuar con el abonado y la efectividad de la navegación. De ahí la importancia de que el SGPV permita el acoplamiento de distintos módulos de reconocimiento.

3.4.4. Servidor Web

El Servidor de Aplicaciones contiene la lógica de la aplicación, es el responsable del manejo de la sesión, de obtener la información solicitada por el Controlador y de su correspondiente envío. Proporciona acceso a contenidos de cualquier origen y de ellos extrae los datos puntuales que necesita el servicio que esté ofreciendo la plataforma. El servidor de aplicaciones podrá estar distribuido físicamente en tantas máquinas como sea necesario para la prestación eficaz del servicio.

El Servidor de Transcodificación está fuera del alcance del presente trabajo, pero se ha incluido en la arquitectura propuesta por cuestiones de completitud del esquema. Puede incluirse en el sistema cuando se desee convertir a formato VoiceXML contenidos de cualquier origen como HTML, WML (Wireless Markup Language) [45] o contenidos no XML (Extensible Markup Language), a partir de su localización. Este servicio proporciona a la plataforma contenidos en VoiceXML, que la plataforma transforma automáticamente a su interfaz vocal para proporcionárselos al usuario.

El Módulo de Personalización se puede incluir en el sistema si es necesario adaptar las páginas VoiceXML antes de ser entregadas al usuario, en esta sección se podrían incluir los filtros de información necesarios de acuerdo con el perfil del abonado que se encuentra conectado con el sistema.

4. Implementación y resultados

Inicialmente se evaluó el funcionamiento de la arquitectura de PublicVoiceXML versión 3.0 cuyo principal avance frente a las versiones anteriores fue el cambio de protocolo de comunicación a Capi y la interacción con Microsoft Speech SDK [38] como motor de síntesis de voz. El problema es que Capi es el protocolo usado para la comunicación con dispositivos ISDN, y en nuestro país estos dispositivos no son tan comunes, por consiguiente se debía pensar en una arquitectura más abierta y de mayor aplicabilidad en

Colombia. Sin embargo se analizó la plataforma para verificar qué tan factible es extenderla para que cumpla los requerimientos del SGPV utilizando técnicas de ingeniería reversa [46]. Para efectuar la prueba de esta versión se utilizó Microsoft Speech SDK 5.1 como motor de síntesis de voz y se obtuvo una versión de prueba de VoIP-CAPI [36], un simulador de dispositivos ISDN para evitar tener que incorporar hardware a la prueba. Después de algunos ajustes en la aplicación su funcionamiento fue correcto, se estableció el canal de comunicación y el script básico de VoiceXML utilizado se procesó sin problema. Posteriormente se realizó un estudio detallado de la arquitectura utilizando mecanismos de ingeniería reversa [46] donde se observó una mejora significativa en el diseño de la plataforma frente a las versiones anteriores, con menor complejidad, una arquitectura más estructurada, con mayor posibilidad de escalamiento y menor acoplamiento entre clases, aunque limitada al protocolo CAPI.

La implementación de la funcionalidad adicional requerida por el SGPV se dividió en cinco etapas, en la primera se hizo la adecuación de la arquitectura para soportar otro tipo de hardware de telefonía, en la segunda se realizó la implementación de los protocolos estándar de comunicación elegidos, SIP y RTP utilizando las bases proporcionadas por Vovida para estos protocolos, en la tercera etapa se efectúo el acoplamiento del nuevo protocolo a la arquitectura y se efectuaron las pruebas de establecimiento del canal de comunicación y transporte de los datos, en la cuarta etapa se efectuaron las adecuaciones para incorporar motores de reconocimiento de voz al sistema y en la quinta etapa se implementó la interacción entre el sistema y el motor de reconocimiento de voz elegido, Microsoft Speech SDK 5.1 en inglés.

Finalizada la aplicación se efectuaron las pruebas informales utilizando un script de VoiceXML sencillo, los motores de síntesis y reconocimiento de voz de Microsoft Speech SDK 5.1 [38] y un simulador de teléfonos SIP en cada uno de los equipos. Se verificó la recepción de varias llamadas simultáneas, el reconocimiento de voz, el reconocimiento de los tonos multifrecuenciales del teléfono, el proceso de síntesis de voz y la reproducción de audio.

Posteriormente se desarrolló un portal de voz básico que describe la interacción entre un usuario y un centro de servicios que le permite consultar el estado de su cuenta, verificar la temperatura en las ciudades registradas en el sistema, consultar los teatros y las películas que están ofreciendo y realizar reserva de tiquetes aéreos, hoteles y carros. Las decisiones de diseño tomadas para el portal de prueba teniendo en cuenta los requerimientos de los portales 3.1 fueron: los métodos de comunicación con el usuario son los comandos de voz y la selección de DTMF para permitirle al usuario distintas maneras de interactuar con el sistema y que exista otra posibilidad de interacción en caso de que el ruido de fondo haga ininteligible la voz del usuario.

En los contenidos estáticos del sistema como saludos de bienvenida o despedida, se utilizan archivos de voz pregrabados para brindar un contacto más real con el usuario, la salida del motor de TTS se empleará en el despliegue de contenidos dinámicos. El formato de los archivos de audio es wav, por su generalidad y porque es compatible con la mayoría de los dispositivos de reproducción de audio. Se seleccionaron gramáticas simples, basadas en palabras que el usuario pueda articular para interactuar con el sistema.

Los diálogos ofrecidos por la aplicación son concisos e indican las alternativas que el sistema tiene para el usuario. En caso de que se presente alguna duda con respecto al funcionamiento de la aplicación, el sistema utiliza diálogos de mayor contenido que le permitirán al abonado tener una mejor comprensión de la tarea que está llevando a cabo, además, tiene como soporte comandos de ayuda, de salida o de cancelación de un comando, que están disponibles durante toda la sesión.

El portal esta diseñado para brindarle ayuda al usuario sin necesidad de que él lo solicite, de manera que si el usuario no responde adecuadamente a las preguntas realizadas por el sistema, se modifiquen los diálogos buscando ser más claros e incluir instrucciones de las respuestas que debe dar el usuario para hacer una solicitud válida.

Para las pruebas con el portal se utilizó la configuración mostrada en la figura 1, de manera que el SGPV se instaló en un servidor, los documentos VXML del portal se instalaron en otro servidor al que tendría acceso a través de la red y las llamadas de los clientes se efectuaron desde otros equipos en la red y desde teléfonos fijos y móviles a través de un gateway SIP. Los resultados obtenidos fueron satisfactorios, se probó la

interacción con los motores de reconocimiento y síntesis de voz, la reproducción de audio, la recuperación de documentos de manera local y a través de http y la comunicación con los usuarios mediante el protocolo SIP.

Finalmente se creó un conjunto de pruebas con el fin de verificar el funcionamiento de la mayoría de las etiquetas de VoiceXML versión 2.0, tomando como base la recomendación de la W3C del 16 de marzo de 2004 [18]. El resultado mostró algunas debilidades en ciertas etiquetas, que deberán tenerse en cuenta en los desarrollos futuros.

5. Conclusiones y Trabajo Futuro

Antes de la aparición de las redes NGN, el desarrollo de aplicaciones de telecomunicaciones implicaba construir plataformas tecnológicas costosas y requería por parte de los desarrolladores poseer grandes conocimientos técnicos acerca de las distintas redes sobre las que iba a funcionar la aplicación, los protocolos requeridos para poder establecer la comunicación y para acceder a la información necesaria, entre otros. Desarrollar aplicaciones para este tipo de redes implica cambiar las arquitecturas propietarias por las basadas en estándares, de manera que los desarrolladores puedan desligarse de la codificación de bajo nivel y del conocimiento de la tecnología base para que se concentren en la creación de servicios.

El concepto innovador de las redes de la siguiente generación o NGN [15][17] unido al resultado obtenido a partir del fenómeno de la convergencia centrado básicamente en la unión del mundo de la voz con el de los datos [5][6][41], ha permitido que surjan nuevas aplicaciones que contemplen las necesidades de la comunidad en la actualidad como la posibilidad de acceder a la información disponible en la Web en el momento y lugar que lo necesiten, han convertido a los portales de voz en la opción ideal para la comunicación entre los usuarios y los sistemas informativos, comerciales, educativos, en general, de cualquier índole, porque no requieren de ningún tipo de capacitación para los usuarios finales y porque además permiten el empleo de medios masivos para el establecimiento de la comunicación como por ejemplo el teléfono. Debido a la acogida en la actualidad de los portales de voz, esta investigación se orientó hacia la búsqueda de una infraestructura que cumpliese con los requerimientos propios de los servicios ofrecidos en estos sistemas.

El cumplimiento del objetivo de realizar un servidor genérico se logró mediante el establecimiento de una arquitectura abierta y modular, que permite utilizar distintos componentes, diferentes motores de reconocimiento y síntesis de voz o extender la funcionalidad del Servidor Genérico para Portales de Voz sin mayor dificultad. La utilización de protocolos estándares para el establecimiento de los canales de comunicación de manera que cualquier dispositivo compatible con el protocolo SIP, hoy en día uno de los protocolos más utilizado por los dispositivos de los entornos VoIP, pueda hacer uso de los servicios ofrecidos en el SGPV. Además, el empleo de VoiceXML como el gestor de diálogo permitiendo que los servicios que se quieran ofrecer en los portales de voz, puedan ser desarrollados en cualquier lenguaje de programación, de la misma forma como HTML establece las etiquetas con las que se puede interactuar con el usuario a través de los navegadores, pero libera al desarrollador del lenguaje de programación utilizado, brindando escalabilidad y flexibilidad al sistema.

Existen diferentes caminos para extender el marco de investigación actual, uno de ellos está enfocado a incluir la funcionalidad para la transcodificación de contenidos de manera que las páginas consultadas por los usuarios puedan incluir documentos WML, HTML o contenidos propietarios como el de los grupos de noticias, o el de los correos, con lo cual se puede ampliar la gama de servicios que podrían prestar los portales de voz. Estos nuevos servicios en especial, podrían requerir la creación del módulo de personalización para filtrar la información de acuerdo al perfil del usuario que ingresa al sistema.

Otro aspecto importante que debe ser investigado si se desea utilizar este servidor para aplicaciones que requieran transacciones de alta confiabilidad como las de *e-commerce* es incluir un módulo de seguridad que incluya mecanismos de confidencialidad, autenticidad, no repudiación e integridad. Teniendo en cuenta la integración del sistema con las tecnologías del habla, se podría pensar en utilizar técnicas avanzadas de reconocimiento de voz como mecanismos de autenticación del usuario.

Finalmente, podría contemplarse la creación de un módulo de reconocimiento y síntesis de voz propios del sistema lo que conllevaría a que no se requiera de componentes externos para el funcionamiento del SGPV.

6. Referencias

[1] Sitio Oficial de OpenVXI: http://www.speech.cs.cmu.edu/openvxi

[2] Sitio Oficial de PublicVoiceXML: http://www.publicvoicexml.org

[3] Sitio Oficial de W3c: http://www.w3c.org

[4] G. Wilde. Call centres find a new voice. International Communications. Marzo, 2002.

[5] E. Miseta. Convergence: An evolution, not a revolution. Business Solutions. Enero, 2004.

[6] P. Meakin. Managing convergente. Industry Viewpoint. Diciembre, 2004.

[7] Data Monitor. The Future of IVR. Diciembre, 2001. Disponible en línea en: http://www.computertelephony.org/uploads/wpapers/3_39.pdf

[8] Aculab. The converged network – what it really means to businesses. Disponible en línea en: http://www.computertelephony.org/uploads/wpapers/548_36.pdf

[9] Thomas C.K. Cheng, Design and Implementation of Three-tier Distributed VoiceXML-based Speech System, Octubre 2001.

[10] K. Singh, A. Nambi y H. Schulzrime, Integrating VoiceXML with SIP services, 2002.

[11] B. Pello, W. Ward, J. Hansen, et al. University of Colorado Dialog System for Travel and Navigation

[12] J. Llisterri, C. Carbó, M. Machuca, et al. El papel de la lingüística en el desarrollo de las tecnologías del habla. Universidad Autónoma de Barcelona. Fundación Española para la Ciencia y la Tecnología, 2003.

[13] Sitio Oficial de Open H.323 Project. http://www.openh323.org

[14] Sitio Oficial de Vovida Project. http://www.vovida.org

[15] Eurescom Project P1109 Next Generation Networks: The services offering standpoint. Disponible en línea en http://www.eurescom.de/secure/projects/P1100-series/P1109/P1109.htm

[16] VoiceXML specification. En línea en: http://www.voicexml.org

[17] C.A. Licciardi, P. Falcarin, Technologies and guidelines for service creation in NGN, In Proc. of the 8th ITU International Conference on Intelligence in Networks (ICIN 2003), Bordeaux, Francia, April 2003. Disponible en: http://www.cercom.polito.it/Publication/Pdf/210.pdf

[18] Voice XML 2.0 W3C Candidate Recommendation. Disponible en línea en http://www.w3.org/TR/voicexml20

[19] P. Cenek. Elvira - a VoiceXML Platform for Research. VoiceXML Forum Volumen 3, Issue 2 – Marzo/Abril 2003. Disponible en línea en: http://www.voicexmlreview.org/Mar2003/features/Mar2003_Elvira1.html

[20] Speech Application Language Tags (SALT). Disponible en línea en: http://www.microsoft.com/speech/evaluation/speechtags/

[21] Sitio Oficial de SALTForum. http://www.saltforum.org/

[22] Protocols at Vovida.org. http://www.vovida.org/protocols/index.html

[23] S.Potter, J. A. Larson, VoiceXML and SALT How are they different, and why?. Speech Technology Magazine. Mayo – Junio 2002.

[24] MIT Spoken Language Systems Group: DARPA Sponsored Research. http://www.sls.csail.mit.edu/DARPAactivities.html

[25] Rudnicky, A., Lunati, J-M., and Franz, A. Spoken Language Recognition in an Office Management Domain, 1991.

[26] The Festival Speech Synthesis System. Centre for Speech Technology Research, University of Edinburgh. Disponible en línea en: http://www.cstr.ed.ac.uk/projects/festival/

[27] Vovida Open Communication Application Library. Session Initiation Protocol (SIP) Stack. Software Version 1.2.1. Disponible en http://www.vovida.org/document/pdf/sip.pdf

[28] RFC 3261. SIP. Disponible en www.ietf.org/rfc/rfc3261.txt

[29] RFC 2705. MGCP. http://www.ietf.org/rfc/rfc2705.txt

[30] RFC 1889. RTP. http://www.ietf.org/rfc/rfc1889.txt

[31] T. Cicic. Real Time Protocols. University of Oslo. Diciembre 2001. Disponible en línea en: http://www.ifi.uio.no/inkomevu/slides/DAG5-6/Real-time%20Protocols.pdf

[32] M. Miller. VoiceXML: 10 Projects to Voice Enable Your Web Site, 2002.
[33] M. Miller. Voice Over IP Technologies: Building the Converged Network, 2002.
[34] R. Ben-Natan, R. Gornitsky, et al. Mastering IBM WebSphere Portal: Expert Guidance to Build and Deploy Portal Applications, 2004.
[35] Sitio oficial de SipCenter. http://www.sipcenter.com
[36] Sitio Oficial de VoIP-CAPI. http://www.ikon-gmbh.com
[37] LibSndFile. Disponible en línea en: http://www.mega-nerd.com/libsndfile
[38] Información técnica acerca de Microsoft Speech SDK 5.1. http://www.microsoft.com/speech/techinfo/apioverview
[39] L. Braña. VUI (Voice User Interfaces) y VoiceXML. Curso de Doctorado Avances en tecnología Web. Mayo 2004.
[40] G. Rojas. Los portales de voz: Internet más natural. Gerencia Tecnológica Informatica. Edición 1.Volumen 1. 2002.
[41] Ashton, Metzler & Associates. The path to Convergence. Febrero, 2004. Disponible en línea en: http://www.ashtonmetzler.com/white_papers_page.htm
[42] P. Bell. SIP Goes Mobile. Americas Telecommunications. Marzo 2003.
[43] T. Davies. SIP: a signal success. Internacional Communications. Diciembre, 2003.
[44] D. Cartwright. Getting computers and telephones to talk. Techworld. Marzo, 2004. Disponible en línea en: http://www.techworld.com/applications/features/index.cfm?featureid=454&Page=1&pagePos=8
[45] T. Wugofski, W. Meng, et al. Beginning WAP, WML & WMLScript. Wrox Press; Primera Edición. Noviembre, 2000.
[46] M. Perry, N. Oskov. Introduction to Reverse Engineering Software. Disponible en línea en: http://www.acm.uiuc.edu/sigmil/RevEng
[47] J. Bannister, P. Mather, S. Coope. Convergence Technologies for 3G Networks: IP, UMTS, EGPRS and ATM. John Wiley & Sons. Febrero, 2004.

Security and Integrity Issues Related To Instant Messaging *

Xiang Dai
Department of Computer Science
University of Houston
Xiang.Dai@Gmail.com

Ernst L. Leiss
Department of Computer Science
University of Houston
coscel@cs.uh.edu

Abstract

The Instant Messaging (IM) service has become a widely utilized real-time communication system, not only among teenagers, but also in corporations. However, IM systems do not have a standard communication protocol. Multiple protocols, such as XMPP and SIMPLE, have been submitted to the Internet Engineering Task Force (IETF). In January 2004, XMPP was approved as a proposed standard protocol.

The essential security issues of IM service are the four S's, stalking, spoofing, spamming, and sniffing. To protect the IM service against the four S's, an IM protocol should have a secure channel, incorporate an authentication mechanism, and support message encryption. In the XMPP protocol, the communication security can be enhanced with three methods. The Simple Authorization and Security Layer (SASL) protocol is used for authentication. The Transport Layer Security (TSL) protocol is required for securing the channel. These two protocols can be utilized together for securing client-to-server and server-to-server communication. End-To-End Object Encryption can be used to encrypt the message.

Besides the communication security, there are other security issues. Messages or transferred files may contain malicious code that can infect the whole network. Although typical anti-virus software does not support the IM protocol, it can nevertheless be used to prevent malicious code. If corporations rely on consumer-grade IM systems, their internal information may be exposed to external networks. Moreover, the administrator of the corporate systems cannot control the security policy of the IM server and prevent others from logging into the server and reading confidential information. A better solution for a corporation is to install an internal IM server. Most IM servers have a peer-to-peer architecture; however, this architecture reduces the ability of administrative communication tracking. As a result, the administrator will not know when confidentiality is breached. Therefore, we conclude that an enterprise-grade solution should apply client-server architecture.

* Support of this work under NSF Grant SFS-0313880 is acknowledged.

1. Introduction

Instant Messaging (IM), or Instant Messaging and Presence, is an internet-based communication service that enables end-users to send messages in a one-to-one and one-to-many fashion in real-time. Generally, the IM client has a chat window, which can display the messages in a conversation, and a main window for a contact list. The contact list can show the username of other IM users, along with indicators of availability. The contact list can be grouped into different categories, such as friends, co-workers, and others. Soon after the Instant Messaging service emerged in the early 90's, it was widely adapted by teenagers. Today, corporations also show interest in it. The research firm Gartner estimated that 70 percent of corporate employees rely on IM while at work [11].

The basic functions of the Instant Messaging service consist of the presence service and the instant message service [1]. The presence service can receive information, store it, and distribute it. This information may be the status of a user, the location of a user, etc. The instant message service can deliver a message from one user to another. Many IM systems on the market support pictures and URLs embedded in the text. The color and font can be personalized. With the user-defined status, IM systems not only can give out the availability of a user, they also can show the activity of the user. For example, with a status indication of "gone to IT department, back in one hour", others can draw inferences about the availability of the user.

Since the IM technology is still evolving, many new functions are being integrated into the systems, such as file transfer, audio/video chat, folder sharing, and application sharing. File transfer allows a user to send files to another user. Folder sharing is to share a user's folder, so that other users can access all files in that folder. Application sharing differs in that only a specific application is shared so that another user can execute the shared application but not access any folder and file in the first user's computer. Audio/video chat is one of the most attractive new functions; it supports audio and/or video in a conversation. With this function, the IM system can even substitute for the conventional telephone, especially in a corporate environment.

2. Instant Messaging Architecture

Instant Messaging systems have one of two architectures: peer-to-peer (P2P) or client-server [3]. Most IM systems on the market have a peer-to-peer architecture, that is, there is no server involved in a conversation. When a user A wants to communicate with another user B, A can retrieve B's address from the server. Then A sends messages directly to B, without transfer by any server. On the other hand, in the client-server architecture, a user cannot send messages directly to other users, but must send them to the server. Then the server redirects the messages to the appropriate users. The benefit of the client-server architecture is that it allows an administrator to track and capture the communication. This is essential for an enterprise, since the enterprise requires higher communication security. However, the disadvantage of the client-server architecture is the higher throughput requirement for the server, especially when users transfer large files. In contrast, in the P2P architecture, messages won't pass through the server. Hence, an IM system based on this architecture can more easily offer functions such as file transfer or folder sharing.

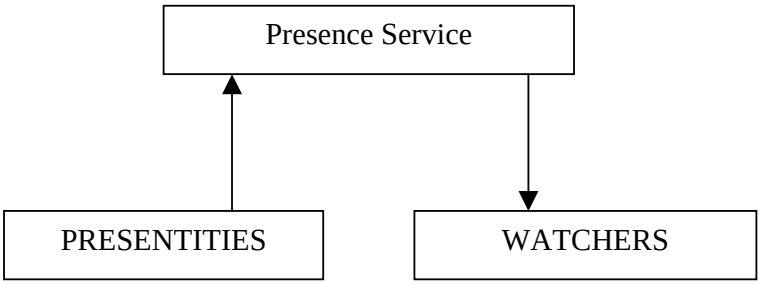
3. Instant Messaging Model and Protocols

The Instant Messaging service has become widely accepted, second only to email; however, it does not have a standard communication protocol. Therefore nearly every IM provider tries to establish its own IM protocol. The following table shows the major IM systems on the market and the protocols they use.

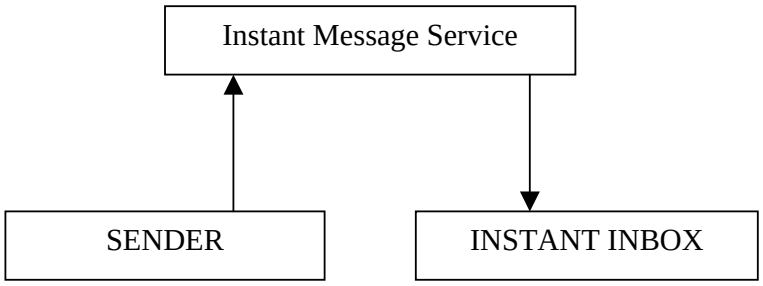
Instant Messaging System	**Protocol**	**Default Port**
AOL Instant Messenger	OSCAR and TOC	5190
Yahoo Messenger	YMSG	5050
MSN Messenger	MSNP	1863
ICQ	OSCAR and TOC	5190
Jabber	XMPP	5222

In response to the growing popularity of Instant Messaging, the Internet Engineering Task Force (IETF) has begun to work on IM standards. The IETF formed the Instant Messaging and Presence Protocol (IMPP) Working Group in 1998. This group has the responsibility for developing a common standard IM protocol. They defined minimum requirements for IM system, which are included in IETF RFCs 2778 and 2779.

The Presence and Instant Messaging service has two basic services: the presence service and the instant message service [1]. The presence service has two kinds of client, PRESENTITIES and WATCHERS. PRESENTITIES provide presence information, which is stored and distributed by the service; WATCHERS can retrieve presence information from the service. WATCHERS can be divided into two types, subscribers and fetchers. A subscriber can get notification from the service when the presence information changes. On the other hand, a fetcher can get presence information only when explicitly required from the service. The model of presence service is as follows:



The instant message service also has two types of client, SENDER and INSTANT INBOX. A SENDER offers instant messages to the service, while an INSTANT INBOX receives messages from the service. When a SENDER sends messages, it gives each message a specific inbox address. The inbox address is associated with a unique INSTANT INBOX. The instant message service tries to deliver the message to the correct INSTANT INBOX. The model of instant message service is as follows:



The IMPP working group defined the model and the minimum requirements for an Instant Message and Presence Service; however, it did not come up with a practical common protocol and leaves this field open. In the defined requirements, the working group only described the requirement of presence service and instant message service, but did not give the definition of other services that have been integrated into current IM systems, such as file transfer and group chat.

4. Instant Messaging Interoperability

Since each Instant Messaging provider utilizes a different protocol and there is no standard protocol by IETF yet, interoperability becomes a real issue. Users have to register an account in an IM system if they want to communicate with someone in that system. It is cumbersome for end-users to install multiple IM client applications and maintain several accounts. Interoperability is not a technical problem, but the consequence of the lack of a standard protocol. Several IM protocols have been submitted to the IETF. Among them, the main competitors are XMPP and SIMPLE.

The SIMPLE Protocol

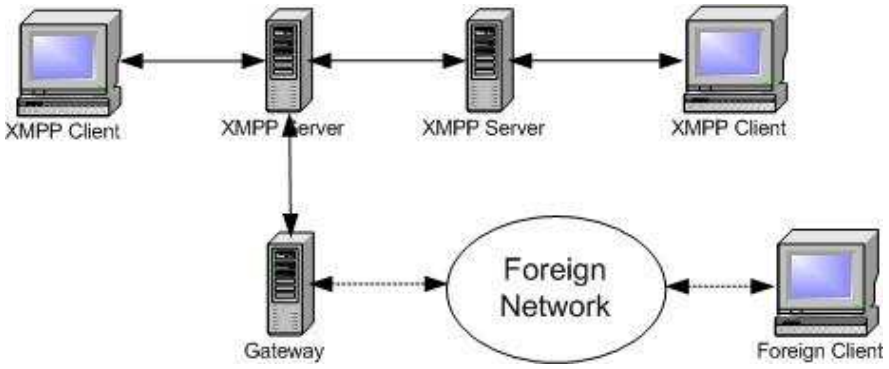
The SIMPLE [14] protocol is the Session Initiation Protocol (SIP) for Instant Messaging and Presence Leveraging Extensions, which is based on the IETF signaling protocol, the Session Initiation Protocol (SIP) [15]. The SIMPLE protocol is proposed for Instant Messaging by extending the SIP protocol, defined in RFC 3261. The SIP protocol is a text-based protocol for initiating interactive communication sessions between different users. The sessions include video, audio, chat, interactive games, and virtual reality. The major software companies that support SIMPLE are Microsoft and IBM. However, the SIMPLE protocol is not finished and remains largely undefined. SIMPLE is very efficient at finding a path between two entity on a network. However, it is ill suited for handing the short IM message exchange, because it requires a handshake to establish each session. Unfortunately, short messages are a big portion of the instant message traffic.

The XMPP Protocol

The Extensible Messaging and Presence Protocol (XMPP) [16] is an XML-based open protocol, which was developed by the Jabber open-source community. The XMPP protocol is intended for instant message exchange and presence information. It includes all basic functionality, defined in RFCs 2778 and 2779. The XMPP protocol has been completed; it has been used by millions of users for five years. The Internet draft of XMPP submitted by the Jabber community consists of four documents: XMPP: core, XMPP: Instant Message and Presence, Mapping the XMPP to Common Presence and Instant Messaging (CPIM), and End-to-End Object Encryption in the XMPP. Of these, the XMPP-core and the XMPP-IM and Presence documents were approved by IETF as proposed standard in January 2004. In this paper, XMPP is used to illustrate what security issues IM systems have, and how IM systems can solve these security issues.

The following figure is the XMPP architecture described in the XMPP-core document [4]. This architecture includes XMPP servers, XMPP clients, and a gateway to foreign networks. The XMPP protocol does not require a particular architecture to support, but generally, it is implemented via client-server architecture, so users do not communicate directly. The XMPP server transfers messages client-to-server, server-to-server, and server-to-gateway. The client connects to a XMPP server over a TCP connection; it can store presence information on the server. Servers are also connected over a TCP connection. The Gateway is used to exchange messages with other non-XMPP systems by translating the XMPP protocol to other protocols.

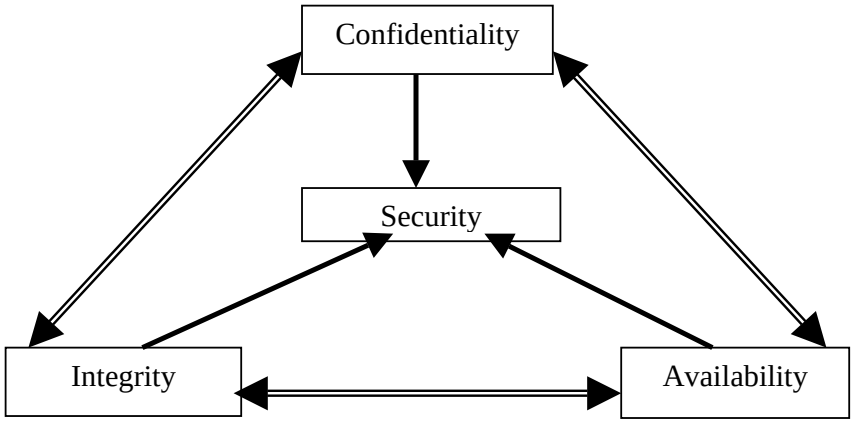
Since the Common Presence and Instant Messaging (CPIM) defined by the IMPP Working Group is the minimum requirement of the Instant Messaging service, the translating at the gateway is based on the XMPP-to-CPIM mapping.



An entity of an XMPP network is an endpoint of the XMPP network, which can communicate with XMPP protocols. It is addressed by a Jabber Identifier or JID. A JID contains of a domain identifier, a node identifier, and a resource identifier. The JID can be used for the identification of an instant message user.

5. Security Issues Related to Instant Messaging

The CIA Model: Information security is becoming more and more important in modern computer environments. To understand and describe information security better, a security model is used to assess the risk of sensitive information. The Confidentiality, Integrity, and Availability (CIA) model [3] is a well-accepted information security model. These three key principles are oriented around the concept of securing data. To offer a secure environment, the three principles must be met by using physical, technical, and administrative controls. Moreover, the three principles must be balanced, so that these principles are equally essential for information security.



Confidentiality is to keep data private, so that data cannot be accessed by unauthorized users. It is most often attacked and most difficult to meet. The usual method to attack network confidentiality is network package sniffing. For example, to attack an IM system, the attacker can "watch" network packages between the user and the server to get username and password. Furthermore, the attacker can "watch" messages exchanged between two users to get confidential information, such as a credit card number. The common technique to ensure confidentiality is

encryption technology embedded in secure connections such as SSL, security guard, password and secure token [3], and so on.

Integrity is to keep data accurate and unchangeable during the storage and the transfer through network. The common methods to achieve integrity are read-only file access, document tracking, and file signatures. In many attacks on network integrity, the attacker intercepts the communication and changes the data before sending them to receiver.

Availability is to keep data available when authorized users require it. A well-known availability attack is DoS (denial of service). File backup and clustering of servers are examples of attempting to ensure the availability of data.

Instant Messaging Communication Security

An Instant Messaging service has to deal with the following four security issues: stalking, spoofing, spamming [1], and sniffing. In stalking, an unauthorized user can get the presence information of other users. The presence information of a user should be fetchable only by those who are granted access by the user. The presence service should prevent unauthorized users from retrieving the presence information. In spoofing, a user pretends to be another user and retrieves sensitive information. The IMPP protocol should provide an authentication mechanism to prevent spoofing. In spamming, a user can send instant messages to anyone no matter whether he or she had agreed to receive them. Different from email service, the instant message service allows a user to send messages only to those who agree to receive them. A user cannot send messages to any instant message inbox address. In sniffing, a network attacker attempts to "watch" the packages transferred on the network in an attempt to find confidential information, such as passwords.

To prevent the four S's, stalking, spoofing, spamming, and sniffing, an Instant Messaging and Presence service should have a secure channel, a strong authentication mechanism, and support message encryption. However, many IM systems on the market do not support a secure channel nor password encryption. Furthermore, almost all IM systems do not encrypt messages exchanged between end-users. The following table shows that the major IM systems support encrypted passwords, but do not encode the message, except AOL Instant Messenger (AIM).

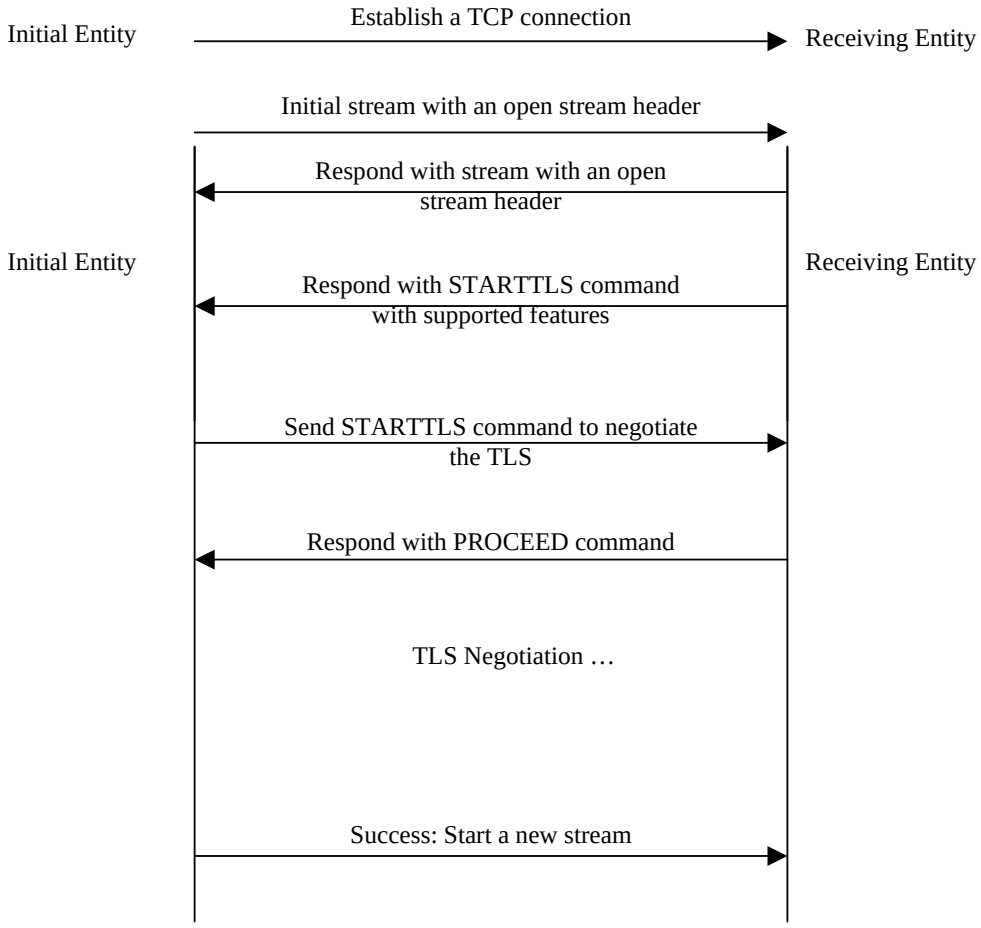
Instant Messaging Systems	**Password Encryption**	**Message Encryption**
AOL Instant Messenger	Yes	Yes
Yahoo Messenger	Yes	No
MSN Messenger	Yes	No
ICQ	Yes	No

Securing a Channel

In the XMPP protocol, the Transport Layer Security (TSL) [6] protocol is utilized for securing the XMPP stream and preventing tampering and eavesdropping [4]. The TLS protocol includes two layers, the TLS Record Protocol and the TLS Handshake protocol, which ensure privacy and integrity of data between two communication entities. The TLS Handshake protocol allows the client and the server to "authenticate each other and negotiate an encryption algorithm and cryptographic keys" [6]. It also provides a secure hash function to compute the MAC, which can be used for data integrity checking. The TLS Record protocol is used for the encapsulation of higher protocols, therefore, the TLS Handshake protocol is independent from the higher protocols and the higher protocols can work transparently on the top of the TLS protocol. In the XMPP

protocol, the Simple Authorization and Security Layer (SASL) protocol can work on top of the TLS protocol to provide authentication. Layered on top of the TCP protocol, the TSL protocol is recommended in the client-to-server and the server-to-server communication in the XMPP network. The minimum rule of the TLS protocol to implement is TLS_RSA_WITH_3DES_EDE_CBC_SHA, a "CipherSuite" defined in TLS 1.0 [6].

When the initial entity attempts to secure an XMPP stream, it first opens a TCP connection and sends an initial stream to a receiving entity. The receiving entity responds to the initial entity and provides the STARTTLS extension as well as other supported features, such as authentication mechanisms. Then the initial entity sends a STARTTLS command to negotiate the TLS and secure the stream. If the receiving entity tries to negotiate the TLS, it sends the PROCEED command back to the initial entity. Otherwise, it responds with an error message. If the negotiation succeeds, the initial entity starts a new stream for communication. If the TLS negotiation fails, the receiving entity will terminate the TCP connection. The steps to establish a secure channel are as follows:

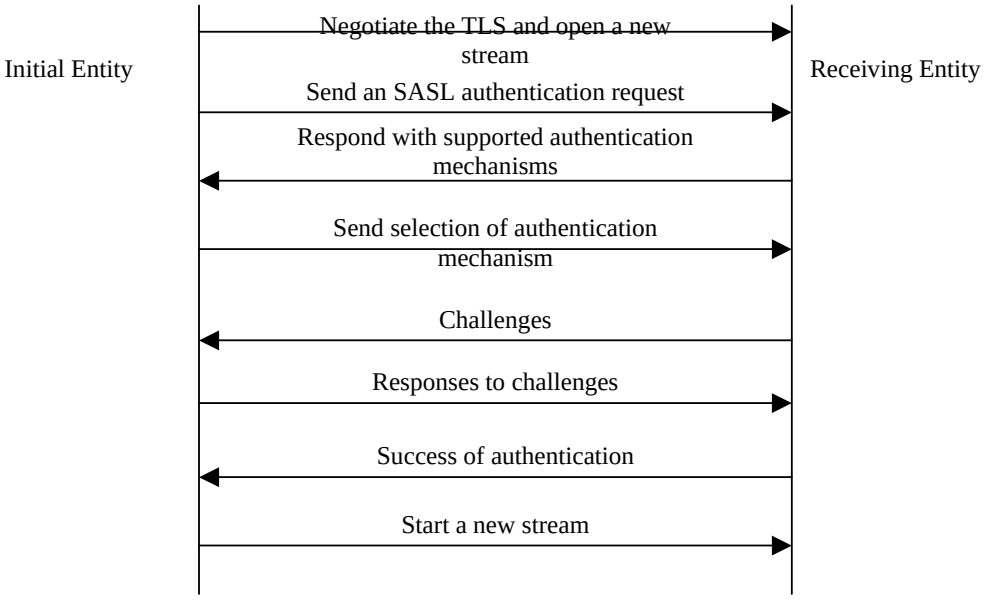


Authentication Mechanism

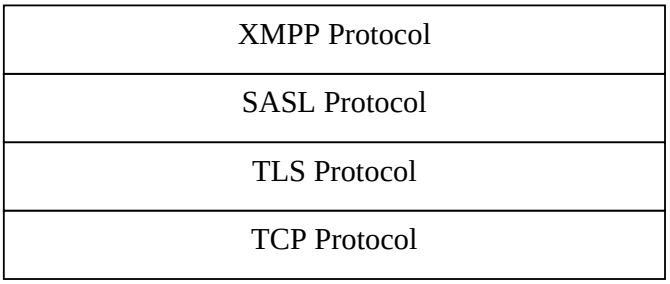
To support authentication in the XMPP protocol, the Simple Authorization and Security Layer (SASL) [7] is required, layered on top of the TLS protocol. The SASL protocol includes a command, which can identify and authenticate a user to a server. It can specify a SASL

mechanism. If the server supports the specified mechanism, it initiates an authentication protocol exchange, which consists of a set of server challenges and client responses. During the authentication protocol exchange, the mechanism performs authentication, transmits an authorization identity from the client to the server, and negotiates the use of a mechanism-specific security layer.

In a communication of the XMPP network, the TLS negotiation happens before the SASL. If the TLS negotiation succeeds, the SASL protocol is used for authentication. First, the initial entity sends a SASL authentication request in an open stream header. The receiving entity responds with a stream header with supported authentication mechanisms. The minimum mechanism of the SASL protocol to implement is DIGEST-MD5, which is an "md5-sess" algorithm of Digest Access Authentication for SASL mechanism [17]. The initial entity selects one authentication mechanism and sends the selection to the receiving entity. Then the receiving entity challenges the initial entity, and the initial entity responds for each challenge. Finally, the receiving entity notifies the initial entity of the success of authentication, and the initial entity opens a new stream to the server. The steps of authentication are as follows:



The client and the server of the XMPP system must implement the TLS and SASL protocol for the client-to-server and the server-to-server communication. The SASL protocol is layered on top of the TLS protocol, while TLS is on top of TCP. The relationship of the protocols is as follows:



End-To-End Object Encryption

While SASL and TLS ensure the security of authorization and channel, the End-To-End Object Encryption [4] in the XMPP provides a method for the sender to sign and encrypt a message, presence information, and XMPP data sent to a specific receiving entity. For interoperability reasons, the encryption should be based on an XMPP-to-CPIM mapping, which translates the XMPP protocol to the CPIM protocol, as defined by the IETF IMPP working group.

The general steps to encrypt messages, presence information and data are as follows:

1) The client generates a CPIM MIME object from the XMPP object. For different objects, the client uses different MIME types. For example, to encrypt a message, the client generates the "Message/CPIM" MIME type. For presence information, the corresponding MIME type is "application/pidf+xml".
2) The client signs and/or encrypts the header and the body of the MIME object.
3) The client embeds the signed and/or encrypted object inside of a XMPP object.

If the XMPP client tries to exchange messages with a non-XMPP client, the messages will be translated by the gateway. The gateway can make the XMPP-to-CPIM mapping. If the message is signed and encrypted, the translation procedure is as follows:

- When the gateway receives an encrypted and signed message from an XMPP client, it removes the XMPP wrapper and sends the MIME object to the non-XMPP network.
- When the gateway receives an encrypted and signed message from a non-XMPP client, it adds an XMPP message wrapper and sends it to the XMPP network.

The encryption operation in XMPP is completed using standard OpenPGP software [8]. The OpenPGP software can be performed in two ways: encrypt and then sign the message, or sign and encrypt. When signing first, the signatories are obscured by the encryption. When encrypting first, the signatories are exposed but the signatures can be verified without decryption. The better way for signing and encryption is that if data are required to be signed and encrypted, the client should sign before encrypting. For quick signature verification, the client should attach a signature to each XML message.

To prevent a replay attack, the encrypted message body includes a timestamp represented as Coordinated Universal Time (UTC), including fractions of a second. The receiving entity will verify the timestamp when it receives messages. If the timestamp is not within five minutes of the current time, or the timestamp is smaller than other timestamps in previous messages, the receiving entity will return an error to the initial entity.

The algorithms to sign and encrypt the message may vary in different implementations of the XMPP protocol; however, all the implementations should support SHA-1 and RSA (PKCS #1 v1.5) with RSA-1 signature algorithms to sign the message, and RSA (PKCS #1 v1.5) and AES-128 encryption algorithms to encrypt the message [9][10].

Other Security Issues Related to Instant Messaging

With the evolution of the Instant Messaging technology, more and more functionalities are integrated into the IM service. Besides the security issues of the IM core, there are more security issues [3] related to these advanced functions.

Malicious Code

File transferring and file sharing are supported by most IM systems, even though the IM specification does not detail these functions. However, a file transferred by IM systems may be infected by a virus or might be a Trojan horse [3]. When other users access the infected file, the malicious code may infect the whole network. Currently, although anti-virus software typically can detect an email virus, it does not support IM protocols. Therefore, the anti-virus software cannot detect and delete malicious code when IM users receive it. But if one scans the computer, or executes an infected file, the anti-software installed on the workstation can detect it. Hence, installing anti-virus software and keeping the virus library up-to-date is a temporary solution for this issue. A permanent solution would be to develop anti-virus software that does support the most widely accepted IM protocols.

Exposing Internal Information to External Networks

Although corporations show interest in the IM service, many IM providers do not offer enterprise-grade solution. Most companies still rely on consumer-grade IM systems within the corporation environment. They install an IM client on every workstation, and use an external "free" IM server to communicate. In consequence, when the corporation users communicate with the server, confidential information may be exposed to the external network [3]. Even if users in the same organization communicate with each other, the messages may still be exposed. As mentioned before, most IM systems, nowadays, do not support message encryption. Therefore, messages may be intercepted.

Moreover, with an external server, the corporate administrator cannot control the authentication mechanism and maintain the confidentiality of user accounts, and that of messages passing through the server. The obvious security issue is that nothing can prevent the external server administrator to read the account information and transferred messages.

Some IM providers have already seen the need for, and begun to implement enterprise-grade IM systems, such as Microsoft Office Real-Time Communication Server and IBM's Sametime system. These products provide internal IM server technology, so that it can be put in place on the corporation network. The internal IM server can allow the administrator to manage the security policy and prevent internal confidential information to be exposed to an external network.

Lack of Communication Tracking

Most IM systems on the market have adopted the peer-to-peer architecture, so the communication between end-users will not pass through the server. This architecture may simplify large data transfer, such as file transfer, but it reduces the ability of administrative communication tracking [3]. The administrator cannot know what kind of information is transferred between users, or if confidential files are transferred to unauthorized user. If administrators cannot track the communication, they cannot know when the confidentiality of data is breached. Therefore, the communication tracking ability is essential to corporations. It is also the requirement of some new US regulations, such as the Sarbanes-Oxley Act of 2002. Hence, for enterprise-grade IM solutions, it is preferred to apply the client-server architecture, not the peer-to-peer architecture. Compared with the peer-to-peer architecture, the client-server architecture can provide the ability to manage, audit, track, and report on the communication.

6. Conclusion

Instant Messaging technology has become a widely adapted communication service, second only to the email service. Not only is it spreading among teenagers, corporations also show interest in

IM service. However, the IM service does not have a standard communication protocol. Different Instant Messaging providers use different protocol to communicate and, currently, IM clients cannot interoperate. The Internet Engineering Task Force (IETF) formed the Instant Messaging and Presence Protocol Working Group and defined the minimum requirements of IM, the Common Presence and Instant Messaging (CPIM), in RFCs 2778 and 2779. However, the IMPP working group did not define a practical protocol and requirements for the common features and left it open to multiple approaches. Among the several approaches submitted to IETF, the Extensible Messaging and Presence Protocol (XMPP) and the SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) are the protocols most likely to be approved. In January 2004, XMPP was approved and became the proposed standard protocol.

The essential security issues of the Instant Messaging and Presence service are the four S's, stalking, spoofing, spamming, and sniffing. Different Instant Messaging protocols use different technologies to address these issues. A few products can protect against all four issues, but most of them can only prevent some of the problems. In the XMPP protocol, the confidentiality and integrity of communication can be enhanced by three methods. The Simple Authorization and Security Layer (SASL) protocol is necessary for authorization. The Transport Layer Security (TSL) protocol is required for secure channels. The two protocols can be utilized together for securing the client-to-server and server-to-server communication. End-To-End Object Encryption can be used to encrypt the message.

The TSL protocol has two layers, the TSL record protocol and the handshake protocol. The record protocol encapsulates the higher protocol so that the handshake protocol can work independently. In the XMPP protocol, the TSL protocol works on top of the TCP protocol, while the SASL protocol is on top of the TSL protocol. Generally, the XMPP client opens a TCP connection to the server. Then it negotiates the TSL protocol. If the TSL negotiation is successful, it starts to negotiate SASL protocol. In TSL and SASL, the server may provide multiple authentication mechanism. The minimum rule for SASL is DIGEST-MD5.

The End-to-End Message and Object Encryption provides a method to encrypt message not only for XMPP networks, but also for non-XMPP networks. Therefore, the encryption procedure is based on the XMPP-to-CPIM mapping. To encrypt a message or an object, they should be translated into MIME objects first. Then the system encrypts a translated MIME object and encapsulates it in an XMPP object. If messages or data are sent to a non-XMPP network, the gateway will remove the XMPP wrapper.

Besides the security issues related to the core function of the IM service, there are some other security issues related to the advanced functions. Messages or transferred files may contain malicious code that can infect the whole network. Although current anti-virus software does not support IM protocols, installing it still can prevent malicious code. If a corporation relies on consumer-grade IM systems, the internal information may be exposed to external networks. Moreover, the administrator of the corporation cannot control the security policy of IM servers and prevent others from logging into the server and reading the confidential information. A better solution for a corporation is to install an internal IM server. Most IM servers have a peer-to-peer architecture; however, this architecture reduces the ability of administrative communication tracking. As a result, the administrator will not know when confidentiality is breached. Therefore, an enterprise-grade solution should apply a client-server architecture.

The Instant Messaging and Presence service is still evolving, and more and more functions are integrated into the service. Functionality makes the IM service powerful and convenient to users. However, it also may result in some security problems. The proposed standard protocol XMPP

only defines the core functions of the Instant Messaging and Presence service. For any other service supported in almost every IM system, such as file transfer, there is no common requirement or protocol. The security issues related to those functions may vary from one IM product to the rest. If the IM technology is utilized in a corporation, the administrator should put in place security controls to ensure the confidentiality, integrity, and availability of corporate data.

References

[1] IETF RFC 2778, "A Model for Presence and Instant Messaging", http://www.ietf.org/rfc/rfc2778.txt
[2] IETF RFC 2779, "Instant Messaging / Presence Protocol Requirement", http://www.ietf.org/rfc/rfc2779.txt
[3] John Stone, Sarah Merrion, "Instant Messaging or Instant Headache?", Communications of ACM, Volume 2, Issue 2, April 2004, pp. 72-80
[4] IETF Internet-Draft, "Extensible Messaging and Presence Protocol (XMPP): Core", http://www.ietf.org/internet-drafts/draft-ietf-xmpp-core-24.txt
[5] IETF Internet-Draft, "End-to-End Signing and Object Encryption in the Extensible Messaging and Presence Protocol (XMPP)", http://www.ietf.org/internet-drafts/draft-ietf-xmpp-e2e-09.txt
[6] IETF RFC 2246, "The TLS Protocol", http://www.ietf.org/rfc/rfc2246.txt
[7] IETF RFC 2222, "Simple Authentication and Security Layer (SASL)", http://www.ietf.org/rfc/rfc2222.txt
[8] Mikko Laukkanen, "Extensible Messaging and Presence Protocol (XMPP)", http://www.cs.helsinki.fi/u/kraatika/Courses/IPsem04s/xmpp.pdf
[9] IETF RFC 2630, "Cryptographic Message Syntax", http://www.ietf.org/rfc/rfc2630.txt
[10] IETF RFC 3370, "Cryptographic Message Syntax (CMS) Algorithms", http://www.ietf.org/rfc/rfc3370.txt
[11] David Greenfield, "IM and the IETF", http://www.networkmagazine.com/shared/article/showArticle.jhtml?articleId=17601120&pgno=1
[12] Joe Hildebrand, "Nine IM Accounts and Counting", Communications of the ACM, Volume 1, Issue 8, November 2003, pp. 44-50
[13] IETF IMPP work group, http://www.ietf.org/html.charters/impp-charter.html
[14] IETF SIP work group, http://www.ietf.org/html.charters/sip-charter.html
[15] IETF SIMPLE work group, http://www.ietf.org/html.charters/simple-charter.html
[16] IETF XMPP work group, http://www.ietf.org/html.charters/xmpp-charter.html
[17] IETF RFC 2831, "Using Digest Authentication as a SASL Mechanism", http://www.ietf.org/rfc/rfc2831.txt

CONSTRUCCIÓN DE UN CIFRADOR BASADO EN UNA PERMUTACIÓN PSEUDO-ALEATORIA

Hernando Castañeda Marín
Universidad de Pamplona, Estudiante de Doctorado Ciencias Aplicadas,
Mérida, Venezuela, 0058
hcastaneda@unipamplona.edu.co
y
Wladimir Rodríguez Graterol
Universidad de los Andes, Doctorado en Ciencias Aplicadas,
Mérida, Venezuela, 0058
wladimir@ula.ve

Abstract

The purpose of this paper is the analysis and development of as practical implementation of a cipher based on the schema proposed by Even and Mansour[1]. The schema corresponds to a block cipher based in the use of a permutation in the encryption operation and in the corresponding inverse function for the decryption process. The key K that consist in two sub keys K_1 and K_2, which are produce by means of a pseudorandom numbers generator with unpredictable inputs and pseudorandom output sequences. To measure the reliability of the permutation of the resulting sequence of the previous XOR operation, hypothesis test were done using the statistic χ^2 for different combinations of the { 0,1 } patterns referenced in those sequences. In order to reach this purpose the Yarrow-160, a visual C++ software from Counterpane System, computational tool was used; extended with the necessary encryption and decryption routines for the proposed cryptosystem based on the Even and Mansour schema. Also a routine was added, that allows doing statistical test by determining the computed values of χ^2 for the different combinatorial patterns of the sequences. This work is an academicals practice to test the strengths and debilities of the Counterpane System software and allows to extend it by means of the construction of a cipher based on a simple permutation that, as practical implementation allows the test of its computational complexity and security. The realization of a cryptanalysis and the realization of different cryptographic attacks will made possible future works in the field of cryptographic and computational security.
Keywords: Cryptographic, Computational Security.

1. INTRODUCCION

Para diseñar un sistema de seguridad se tiene que seleccionar los servicios de seguridad que hayan sido previstos y luego seleccionar los mecanismos que se implementarán por ejemplo: La encriptación, las firmas digitales, el control de acceso, la integridad de los datos y la autenticación.

La encriptación es una función matemática, los algoritmos de encriptación deben tener la propiedad de que los datos originales puedan ser recuperados a partir de los datos encriptados, si el valor de la clave utilizada es bien conocido. Las entradas al algoritmo de encriptación son referidas como texto plano, la salida del algoritmo es denominada texto cifrado o criptograma.

Las claves deben ser mantenidas en secreto, el criptograma estará comprometido si un intruso puede deducir la clave analizando la información pública disponible. La fortaleza de un criptosistema es medida por la dificultad, usualmente medido en el número de operaciones elementales, para determinar dicha clave.

Hay dos clases de criptosistemas, simétricos y asimétricos, llamados también de encriptación pública. Un criptosistema simétrico utiliza la misma clave en ambos extremos de un canal de comunicación, o una clave que es fácilmente derivada de otra, en caso de hacer uso de dos claves. En los criptosistemas asimétricos cada usuario tiene dos claves, una pública y una privada, la cual no es revelada.

La encriptación presenta algunas limitaciones prácticas visibles como son: Los mensajes de entrada a los algoritmos no cuentan con protección, el intercambio de claves, se realiza sobre los mismos canales en que se transmiten los datos, la encriptación no provee protección contra las modificaciones, esta puede ser detectada incluyendo como parte de los datos encriptados un número de bits de chequeo.

La metodología empleada durante la investigación cumplió con los siguientes pasos: Una revisión bibliografica exhaustiva que terminó con una sustentación publica del anteproyecto, una revisión bibliográfica del artículo de Even y Mansour [1] y una revisión computacional de criptosistemas y generadores de números seudo-aleatorios como requisitos previos para una sustentación pública de un avance de tesis; en la fase final se desarrollo el criptosistema enriqueciendo el software del PRNG Yarrow-160 y la aplicación de prueba estadística con el afinamiento del presente informe final.

2. EL CIFRADOR DE EVEN Y MANSOUR

En [1] Even y Mansour proponen la construcción de un cifrador de bloques usando únicamente una permutación seleccionada en forma aleatoria.

El propósito del presente trabajo es desarrollar y analizar una implementación práctica del cifrador de Even y Mansour. Adicionalmente los autores asumen que la permutación se supone que es de acceso público (Como una caja negra), y que cualquiera que desee atacar el cifrador tiene acceso a ella. Los autores "prueban" la seguridad del cifrador propuesto bajo la suposición de que la permutación es aleatoria, o al menos seudo- aleatoria, y que la única vía de accesar a través de una caja negra.

La figura 1 ilustra el esquema de Even y Mansour donde M =mensaje, F =permutación $\oplus$ = XOR (bit a bit) $K = \{k_1, k_2\}$ =Clave, C = criptograma.

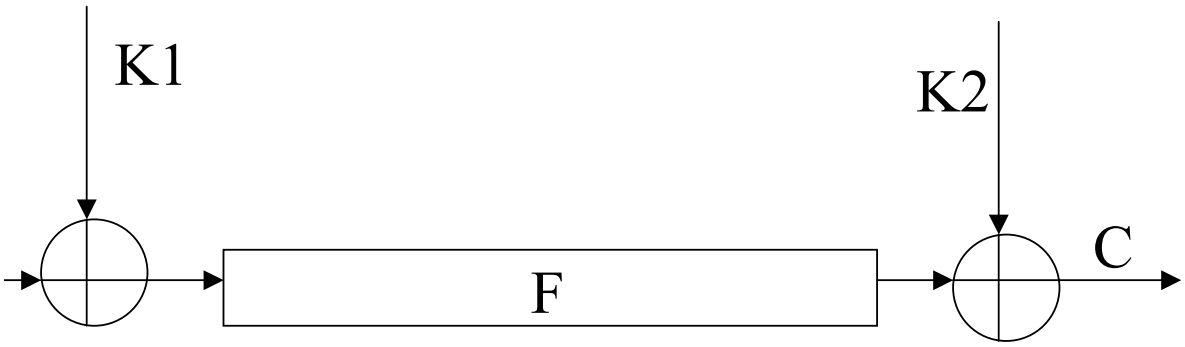


Figura 1. Esquema de Even Y Mansour.

Se asume que M y C son palabras binarias de longitud n, esto es que M y C son elementos del conjunto
También se asume que $F(x)$ es una permutación de $\{0,1\}^n$, y se denota su inversa por $F^{-1}(x)$, y que es "fácil" computar $F(x)$ o, $F^{-1}(x)$, ya sea directamente o usando una caja negra, la cual se denominara "oráculo".

La clave K consiste en dos subclaves k_1 y, k_2 cada una seleccionada en forma aleatoria del conjunto $\{0,1\}^n$.

Se asume que K es conocida solamente por las partes autorizadas, esto es el emisor y el receptor, y que esta es usada para encriptar mensajes por un largo periodo de tiempo.
El proceso de encriptación puede ser descrito por la siguiente ecuación:

$C = E_K(M) = F(M \oplus K_1) \oplus K_2$, Donde C es el criptograma correspondiente a $\{0,1\}^n$, y $\oplus$ denota la operación XOR (OR exclusivo), bit a bit. El proceso de descifrado puede ser descrito por la siguiente ecuación:

$$M = D_K(C) = F^{-1}(C \oplus K_2) \oplus K_1$$

2.1 DEFINICIONES DE SEGURIDAD Y SUS RELACIONES

En su artículo [1], Even y Mansour modelan al "adversario" como alguien que puede apoderarse por un tiempo determinado del cifrador para cifrar y descifrar mensajes escogidos por él puede usar las permutaciones F y F^{-1}, como cajas negras pero que no tiene acceso a la clave K. Even y Mansour modelan la situación anterior usando oráculos, los cuales responden preguntas de naturaleza particular.
El "problema del rompimiento "CP, es definido como un intento (por un adversario) de descifrar un criptograma dado $C_0 = E_k(M_0)$, sin un conocimiento de la clave K. Se asume que el algoritmo usado por el adversario tiene acceso a los siguientes oráculos:

1. El $F - oráculo$: Dado $x \in \{0,1\}^n$, el oráculo suministra $F(x)$.
2. El $F^{-1} - oráculo$: Dado $x \in \{0,1\}^n$, el oráculo suministra $F^{-1}(x)$.
3. El $E - oráculo$: Dado $M \in \{0,1\}^n$, el oráculo suministra $E_k(M)$.
4. El $D - oráculo\ \ Co - restringuido$: Dado $C \in \{0,1\}^n$, tal que $C \neq C_0$, el oráculo suministra $D_K(C)$.

En otras palabras se supone que el adversario tiene acceso a una colección arbitrariamente grande de pares (M, C), donde C es el criptograma correspondiente a M, generados usando la clave K, la cual no es conocida por el adversario y además, que la permutación F es públicamente accesible como una caja negra.
El algoritmo del adversario es exitoso si éste produce $M_0 = D_k(C_0)$. La probabilidad de éxito del algoritmo es la probabilidad de que sea escogida en forma aleatoria C, (con distribución uniforme) y el algoritmo produzca M_0.

El problema del rompimiento es también conocido como $ataquepormensaje \big/ criptogramaescogidos.$

Ahora se considera el problema de la falsificación existencial, EFP.
El adversario tiene acceso a cuatro oráculos: el $F - oráculo$, $F^{-1} - oráculo$, $E - oráculo$, $D - oráculo$ no restringido. Este último es definido como sigue: dado cualquier $C \in \{0,1\}^n$, el oráculo suministra $D_k(C)$.

El objetivo del adversario, en el problema EFP, es encontrar un nuevo par (M', C'), con $C' = E_K(M')$., De esta forma el adversario puede producir un criptograma C' valido para mensajes con "sentido" $M' \neq M_j$ sin conocer K. Así el adversario puede enviar mensajes cifrados como si fuera un usuario autorizado.

Definición: sea f una función de los enteros positivos hacia $[0,1]$.

Se dice que f es de forma polinomial despreciable si para todo polinomio hay un n_0, tal que si $n \succ n_0$ entonces $f(n) \prec \frac{1}{p(n)}$. [1]

[1] Comparación asintótica entre dos funciones en el intervalo $[0,1]$ [5]

Se asume que el adversario emplea un "algoritmo randómico",[2] cuyo tiempo de corrida es de grado polinomial acotado en n, para resolver los problemas CP y EFP.

Se dice que un problema es duro, si, para todo algoritmo randómico, la probabilidad de éxito es polinomialmente despreciable. La probabilidad es tomada sobre todas las escogencias hechas en el diseño del cifrador (o sistema) , esto es las escogencias de los F, las claves escogidas por el usuario, y los lanzamientos de moneda realizados en el algoritmo del adversario.

Even y Mansour reducen EFP a CP, esto es, ellos muestran que la existencia de un ataque exitoso para el problema CP implica la existencia de un ataque exitoso para el problema EFP.

Teorema [Corolario 3.2] Si todo algoritmo, con tiempo de corrida polinomial, para el problema EFP tiene una probabilidad de éxito despreciable, entonces todo algoritmo, con tiempo de corrida polinomial, para el problema CP tiene una probabilidad de éxito despreciable.

El resultado principal del artículo de Even y Mansour es el siguiente:

Teorema **[1, *Teorema* 4.4]** La probabilidad de que un algoritmo randómico A resuelva el problema EFP, cuando F y K son escogidas aleatoria y uniformemente, está acotada por:

$O(\frac{lm}{2^n})$ Donde l es él número de preguntas a los oráculos $\frac{E}{D}$, y m es el número de preguntas a los oráculos $\frac{F}{F^{-1}}$.

Como corolario del resultado anterior se obtiene lo siguiente:

Corolario Si F es escogida en forma aleatoria, entonces todo algoritmo randómico con tiempo de corrida polinomial, tiene una probabilidad de éxito despreciable. Así desde un punto de vista teórico, la probabilidad de "romper" el cifrador de Even y Mansour es despreciable, si F es escogida aleatoriamente.

Si La Familia de permutaciones F son seleccionadas seudo-aleatoriamente y aunque cualquier adversario tiene acceso a los cuatro oráculos, se garantiza que éste no llegará a las entrañas de la caja que implementa F.

La anterior aserción justifica el siguiente teorema **[2, *Teorema* 5.1]** Si F es seleccionado en forma aleatoria y k es seleccionado uniformemente, entonces para todo algoritmo limitado en grado polinomial para solucionar el problema EFP, la probabilidad del evento es de grado polinomial limitado

2.3 VENTAJAS EN SU CONSTRUCCIÓN

Los análisis realizados por Even y Manssour [1] y confirmadas por Daemen [4] acerca de las características sobresalientes en la construcción de su cifrador de bloque y que específicamente son: la consideración establecida para el operador de encriptación interno F el cual es fijo y público; y la formalización de su esquema de encriptación. Adicionalmente la demostración sobre la efectiva longitud de la clave $n - lg\ l - lg\ m$ en donde al adversario se le permite realizar l llamadas al oráculo de encriptación y desencriptación y m llamadas al oráculo de la permutación $F\ o\ F^{-1}$.

Como ejemplo de motivación de esta implementación está la construcción de DESX introducida por Rivest y que permite proveer información económica acerca de la clave en DES.

[2] Los algoritmos aleatorios se clasifican de acuerdo a la probabilidad de que retornen una respuesta correcta Pág. 62 [5]

3. CONSTRUCCIÓN DEL CIFRADOR

Inicialmente se explicarán los mecanismos del generador de números seudo-aleatorios PRNG pero el propósito de utilizar Yarrow-160 para generar las claves se justifican por ser una utilidad para la acumulación de entropía.

Un número aleatorio es un número que no puede ser predecible para un observador antes que éste sea generado. Si un número está en el rango $0...2^{n-1}$, un observador no puede predecir el número con probabilidad superior a $1/2^n$.

Un PRNG contiene un estado interno que es usado en determinados instantes para generar salidas seudo-aleatorios. Este estado guarda confidencia y controla significativamente los procedimientos.

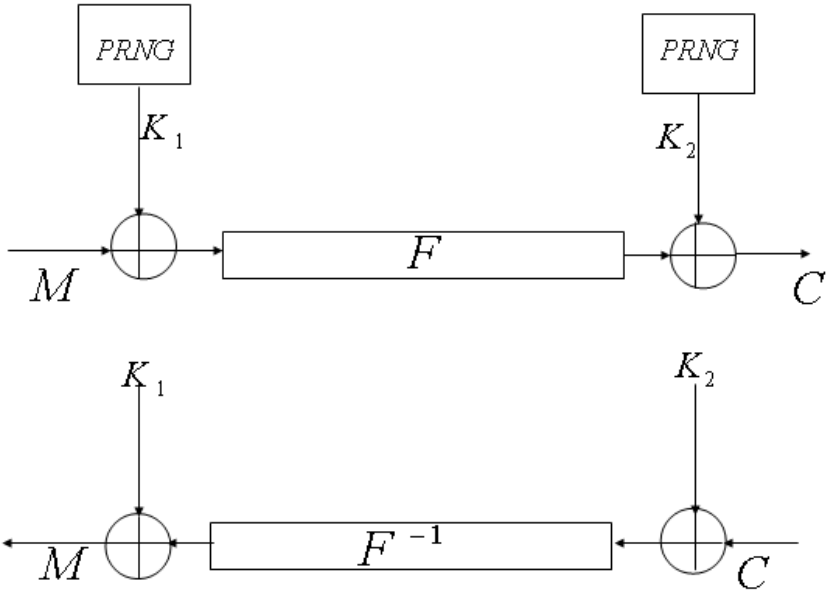


Figura 2. Esquema de implementación

Donde M =mensaje, F =permutación $\oplus$ = XOR (bit a bit),PRNG generador de números pseudo-aleatorios, $K = \{k_1, k_2\}$ =Clave, C = criptograma.

3.1 HERRAMIENTA CRIPTOGRÁFICA

Un generador de números seudo-aleatorios es un mecanismo criptográfico para procesar entradas impredecibles y generar salidas seudo-aleatorias. Si se diseña, implementa y utiliza apropiadamente, cualquier adversario con enormes recursos computacionales no será capaz de predecir una secuencia de la salida del PRNG.

Un adversario puede comprometer el estado interno mediante el uso de un modelo muy bueno que produzca valores impredecibles, utilizando las muestras de entrada del PRNG y con un gran manejo de poder computacional, trate de adivinar el estado interno del PRNG.

El PRNG posee cuatro componentes principales:

1. Un acumulador de entropía que colecciona muestras desde las fuentes de entropía y las almacena en dos piscinas.

2. Un mecanismo de resiembra el cual sirve periódicamente de semilla en la generación de la clave, con nueva entropía en las piscinas. El proceso de combinar la clave existente y las nuevas muestras dentro de una nueva clave, recibe el nombre de resemillar.
3. El mecanismo de generación que genera salidas de claves del PRNG.
4. El control de resiembra, el cual se determina cuando una resiembra sé esta realizando.

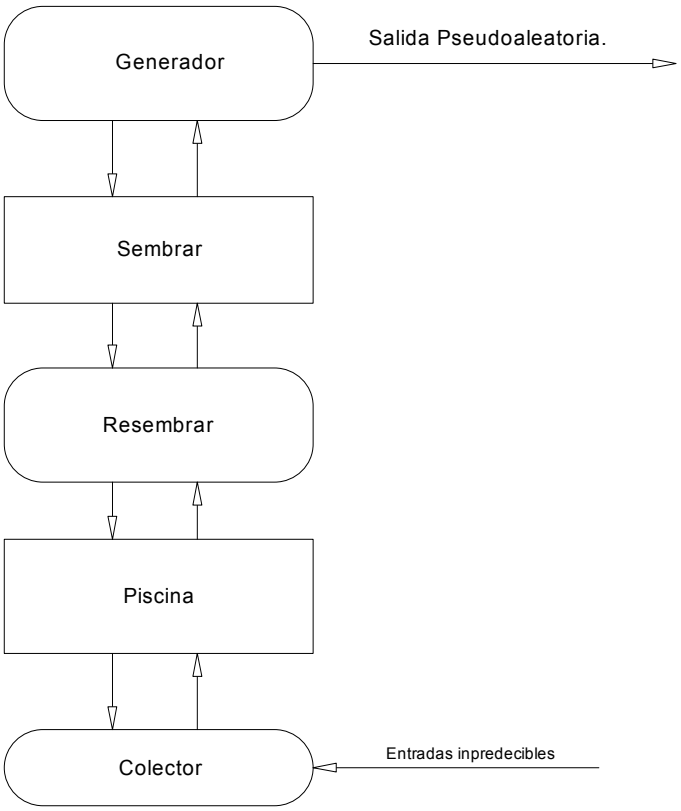


Figura 3 Generalización de un PRNG con resembrado periódico

A continuación se especifica el rol de cada componente en el diseño del PRNG y los requerimientos para cada componente en términos tanto de seguridad como de desempeño. También se especifica la forma como cada componente puede interactuar con los otros.

3.1.1 ACUMULADOR DE ENTROPÍA (COLECTOR).

La acumulación de entropía es el proceso de inicialización por el cual el PRNG adquiere un nuevo estado interno no adivinable. Durante la resiembra es indispensable que la entropía se acumule completamente desde las muestras, con el fin de evitar los ataques iterativos de adivinación [6]. Es importante también que se estime correctamente la cantidad de entropía que se tiene que coleccionar y ésta tiene que ser muy grande. El mecanismo de acumulación de entropía puede también resistir el ataque de selección de entrada [6],

En Yarrow, la entropía de las muestras es coleccionada en dos piscinas, cada una en contexto de mezcla. Las dos piscinas son una piscina rápida y una piscina lenta. La piscina rápida proporciona resiembras frecuentes de la clave, con el propósito de asegurar en lo posible que la clave comprometida tenga una corta duración y la medición de la precisión de los estimativos de la entropía de cada fuente. La piscina lenta proporciona pocas y conservativas resiembras.

3.1.2 MECANISMO DE RE-SIEMBRA.

El mecanismo de resiembra conecta al acumulador de entropía con el mecanismo de generación, esto es posible cuando el control de resembrado determina que esto se requiere, entonces el componente de resembrado modifica la clave y lo hace utilizando el mecanismo de generación, con información de una o ambas piscinas y que son actualizadas por el acumulador de entropía, de tal forma que el objetivo sea que la clave o la piscina sean desconocida por el adversario, después del resembrado.

El mecanismo de resiembra genera una nueva clave K para el generador desde la piscina del acumulador de entropía y la clave existente. El tiempo de ejecución del mecanismo de resiembra depende de un parámetro $P_t > 0$. Este parámetro puede estar fijado para la implementación o ser dinámicamente ajustable.

El proceso de resiembra consta de las siguientes etapas:

1. El acumulador de entropía computa el hash sobre la concatenación de todas las entradas dentro de la piscina rápida, El resultado se llama v_0
2. Se asigna $v_i := h(v_{i-1} \mid v_0 \mid i) para\ i = 1,...,t.$
3. Se asigna $K \longleftarrow h'(h(v_{P_t} \mid K), k).$
4. Se asigna $C \leftarrow \mathrm{E_k}(0)$.
5. Inicializa a cero todos los acumuladores de estimativos de entropía en el acumulador de entropía.
6. Limpia la memoria de todos los valores intermedios.
7. Si un archivo de semilla está en uso, los próximos $2k$ bits de salida del generador están escritos para el archivo de semilla, entonces sé sobre-escribe el valor presente.

En la etapa 1 se recolecta la salida del acumulador de entropía. En la etapa 2 se utiliza una fórmula iterativa de longitud P_t y se hace la resiembra, que es computacionalmente costosa. En la etapa 3 se utiliza una función hash h y una función h', la cual crea una nueva clave K a partir de la actualizada y un nuevo valor de entropía. La etapa 4 define un nuevo valor para el contador.
La función h' esta definida en términos de h. Para computar $h'(m,k)$ se construye así:

$$s_o = m$$
$$s_i = h(s_0 \mid ... \mid s_{i-1})\ i = 1,...$$
$$h'(m,k) := first\ \mathrm{k\ bits\ of}\ (\mathrm{s_0} \mid s_1 \mid ...)$$

Esto es efectivamente una función conocida como "adaptador de tamaño" que convierte una entrada de cualquier longitud en una salida de longitud específica.

3.1.3 MECANISMO DE GENERACIÓN.

Se tiene un valor C de N-bits, para generar los próximos bloques se incrementa C y se cifra usando la clave, Para generar el próximo bloque de salida, se ejecutan las siguiente ecuaciones:

$$C \longleftarrow (C+1) \bmod 2^n$$
$$R \longleftarrow E_K(c)$$

Donde R es el próximo bloque de salida y K es la clave actualizada del PRNG.

Sí la clave está comprometida en cierto punto en el tiempo, el PRNG no debe producir lentamente muchos rendimientos de salida anteriores a las que fueron generadas antes del compromiso.

Es claro que este mecanismo de generación no es inherentemente resistente a una clase de ataque. Por esta razón se investiga cuántos bloques tienen rendimiento. Una vez que se alcanza algún límite P_g (parámetro de un sistema de seguridad $1 \le P_g \le 2^{n/3}$), se genera k bits de salida del PRNG y se usan estos como nueva clave.

$$k \longleftarrow \textit{proximo}\ \text{k bits de la salida del PRGN}$$

Se llama esta operación un generador de puerta. Nótese que todo no es una operación de resiembra y ninguna nueva entropía se introduce dentro de la clave.

Con el interés de guardar un diseño conservativo extremo, el máximo número de salidas del generador entre resiembras está limitado a $min\left(2^n, 2^{k/3} P_g\right)$ bloques de salida de n-bits. El primer término en el mínimo previene el valor de C en el ciclo.

El segundo término hace que sea extremadamente improbable que K repita el mismo valor dos veces. En la práctica P_g puede ser un conjunto más pequeño que éste, por ejemplo, en el orden de minimizar el número de salidas que pueden ser aprendidas por retro-alimentación.

3.1.4 CONTROL DE RESIEMBRA.

El mecanismo de control de sembrado se presenta teniendo en cuenta las siguientes consideraciones. Frecuentemente el resembrado es deseable, pero probablemente es vulnerable a un ataque de adivinación iterativo [6]. Un resembrado poco frecuente, permite actuar al adversario, siempre que tenga comprometida la clave con más información.

Existe control en los estimativos de entropía para cada fuente, así, como es que llegan las muestras a cada piscina. Cuando cualquier fuente en la piscina rápida ha pasado el valor del umbral, entonces se resiembra desde la piscina rápida. En muchos sistemas, se puede esperar que esto pase muchas veces en una hora.

Cuando cualquier K de las n fuentes tiene un alto umbral en la piscina lenta, entonces se resiembra desde la piscina lenta. Este es un proceso mucho más lento.

Para Yarrow –160, el umbral para la piscina rápida es de 100 bits y para la piscina lenta es de 160 bits. Al menos dos diferentes fuentes están sobre 160 bits en la piscina lenta antes de la resiembra en la piscina lenta[6]. (Esto se facilita para diferentes ambientes, los ambientes con tres fuentes de entropía rápida son buenos y razonables).

Él módulo de control de resiembra determina cuándo la resiembra está siendo realizada. Ocurre cuando alguna aplicación explícita pregunta por la operación de resiembra. Esto se usa raramente y sólo por aplicaciones que generan valores muy altos de aleatoriedad. El acceso a funciones de resiembra explícita puede ser restricto en muchos casos.

Las Re-siembras[3] periódicas ocurren automáticamente. La piscina rápida se usa para la resiembra, sin embargo cualquiera de estas fuentes tiene una entropía estimada sobre algún valor de umbral. La piscina lenta es usada para resembrar, sin embargo, al menos dos de las fuentes tienen estimativos de entropía antes que otros valores de umbral.

3.2 ESTIMACIÓN DE LA ENTROPÍA

La estimación de la entropía es el proceso de determinar, cuánto trabajo puede tomar un adversario para adivinar el contenido actual de las piscinas.
La implementación será costosa por la determinación de sus fuentes. Las fuentes no están cerradamente encadenadas, o exhiben cualquier correlación significativa.

La entropía de cada muestra es medida en tres formas:

[3]Entropía: magnitud que mide la información contenida en un flujo de datos, es decir, lo que nos aporta sobre un dato o hecho concreto [Wikipedia]

1. El programador suministra un estimativo de entropía en una muestra cuando escribe la rutina para coleccionar datos de la fuente. Así el programador puede enviar en una muestra, un estimativo de 20 bits de entropía.

2. Por cada fuente se utiliza un estimador estadístico de la muestra. Esta prueba se hace para descubrir situaciones anormales debido a que la muestra tiene una entropía muy baja.

3. Para determinar la máxima amplitud muestral, "densidad" se opera con la longitud de la muestra en bits y multiplicando por algún factor constante menor de uno, esto con el propósito de obtener un estimativo máximo de entropía en la muestra. En Yarrow-160 se usa un multiplicador de 0.5.

Se usa el valor más pequeño de estos tres estimativos como entropía de la muestra en cuestión.

3.3 GENERANDO SALIDAS SEUDO-ALEATORIAS

El mecanismo de generación puede tener las siguientes propiedades:

1. Resistencia al ataque cripto analítico.
2. Eficiente.
3. Resistente a la retroalimentación después de una clave comprometida
4. Capacidad de generar secuencias muy largas de rendimiento firme sin resembrar.

Hay dos puntos del plan principal que deben ser tenidos en cuenta con este generador seudo-aleatorio. El primero tiene en cuenta tanto la criptografía, como la calidad de sus entradas por su seguridad y calidad de los datos. El segundo consiste en que el mecanismo del rendimiento y piscinas de entropía son tan distintos como es posible.

4. FUNCIONAMIENTO DEL CRIPTOSISTEMA

4.1 Ejemplo que ilustra el proceso de cifrado y descifrado.

En el próximo párrafo se quiere realizar un ejemplo hipotético que demuestra como funcionan los procesos de cifrar y descifrar bloques del archivo de texto o del texto cifrado respectivamente. Esto es simplemente la abstracción del esquema propuesto por Even y Mansour [1] que es la base de la implementación pero enriquecido con la Funcionalidad PRNG para generar las claves.

4.1.1. PROCESO DE CIFRADO.

Se tiene en cuenta la expresión matemática que formaliza este proceso

y teniendo en cuenta lo anterior se establecen tamaños hipotéticos para la clave y el bloque y se generan secuencias de dicho tamaño para las posibles claves o sea k_1 y k_2.

Se considera un tamaño de bloque por ejemplo de 8 bits sea $b_1 = 10110111$ correspondiente al bloque texto y el PRNG genera una clave también de 8 bits por ejemplo $k_1 = 11101011$.

Sea $b_1 \oplus k_1 = 10110111 \oplus 11101011 = 01011100$. A partir de esta operación se seleccionan en forma aleatoria dos números diferentes, en este caso entre 1 y 8 por ejemplo que sean 2 y 7, se intercambian los bits de las posiciones 2 y 7, de la secuencia 01011100 Esto permite producir una de sus permutaciones, que en este caso es $P_1 = 00011110$. Ahora, utilizando PRNG, generamos una nueva clave, por ejemplo, $k_2 = 10101010$ .Sea $p_1 \oplus k_2 = 00011110 \oplus 10101010 = 10110100$ que corresponde al bloque cifrado. La clave compuesta para el proceso de descifrar es $k_1 = 11101011$ y $k_2 = 10101010$ y los índices de la permutación, buscando como medio el uso de un canal seguro.

4.1.2 PROCESO DESCIFRADO.

La descripción del proceso de descifrado del algoritmo criptográfico es sencilla. Consiste en sustituir las transformaciones utilizadas en el cifrado por sus inversas e invertir el orden de aplicación de dichas transformaciones o funciones matemáticas.

Teniendo en cuenta esto el proceso de descifrado es descrita, por la siguiente ecuación: El bloque cifrado es $Q_1 = 10110100$ ahora con la clave del proceso anterior $k_2 = 10101010$ se opera $Q_1 \oplus k_2 = 10110100 \oplus 10101010 = 00011110$, ahora se opera con la permutación inversa intercambiando los bits de la posición 2 y 7 de la secuencia 00011110 se produce $p_2 = 01011100$, ahora operando en el ámbito de bit se llega a $p_2 \oplus k_1 = 01011100 \oplus 11101011 = 10110111$.

Se puede observar que se llega a la misma secuencia con la que se inicio el proceso de cifrar el bloque del archivo de texto.

El proceso experimental se realiza con bloques y claves de 120 bits.

4. RESULTADOS

El proceso investigativo del presente trabajo se inicia con el conocimiento del esquema propuesto por Even y Mansour, y fue necesario entender qué es un cifrador de bloque y cómo una sola permutación es seleccionada en forma aleatoria y utilizada, tanto en el proceso de cifrado como en el descifrado. Con base en los resultados, se puede ver inmediatamente, que la clave que modifica la permutación es muy sencilla y fácil de implementar.

Dentro del procedimiento de aplicación, se realiza a continuación la ejecución del programa que cifrara el archivo de entrada.
Para comprobar la terminación de este trabajo de grado, también se implementó el programa que permite descifrar el archivo presentado anteriormente, produciendo el siguiente resultado.

```
// File choreographer.kPp  (from Karel++ textbook
//  by Bergin,et.Al. see p. 56(
// Run this on file "harvest.Wld"

class CHoreographer : Ur_Robot
{
  □ ur_Robot Lisa(4,2, East, 0);  // the 1st heLper robot
   □ur_Robot Tony(6,2, East, 0);  // the wnd helper robot

    void harvEst();
    void harvestARow()□
    void harvestCorner();
    void move()□
    void pickBeeper();
   □void turnLeft(            ;
    void tuRnOff();
};

void ChoreograPher::harvest()
{
    harvestARow();
    tUrnLeft();
   □move();
    tUrnLeft();
   □harvestARow((;
}
```

```
task
{  ChoReographer KareL (2,2, East, 0       ;
   Karel.hArvest();
   Karel.turnOff()□
}
```

Se puede observar que los archivos presentan características similares para un formato dado, pero no podemos afirmar que son dos copias exactamente iguales para todo tipo de formato de edición.

Finalmente, para demostrar la hipótesis sobre el uso de una sola permutación en el esquema propuesto y el uso de un generador de números seudo-aleatorios se realizó un programa que implementa cinco pruebas estadísticas que son utilizadas generalmente para determinar si una secuencia binaria, como son las permutaciones generadas en esta implementación posee características específicas de secuencias seudo-aleatorias verdaderas, como probablemente se ha declarado.

Estos valores para el estadístico χ^2 reciben el nombre de valores calculados; y comparándolos contra los valores críticos de χ^2 en la tabla 5.1 y 5.2 en [5] para secuencias de tamaño n=120, nos permitirá comprobar o rechazar la hipótesis de esta investigación.

En la tabla a continuación para cada prueba estadística muestra la formula del estadístico χ^2 con los respectivos grados de libertad y los valores calculados con el programa de test estadístico[ver apéndice D] y los respectivos valores críticos con un nivel de significancia $\alpha = 0.05$.

La tabla nos permite establecer que para la secuencia mostrada anteriormente las pruebas de frecuencia y la prueba serial y se quedan las pruebas poker, test y correlación.

Prueba estadística	Formula para chi-cuadrado	G.L	χ^2 calculada	χ^2 Estimada
Frecuencia	$X_1 = \frac{(n_0 - n_1)^2}{n}$ $n_0 = 59$ y $n_1 = 61$	1	0.033	3.8415
Serial	$X_2 = \frac{4}{n-1}\left(n_{00}{}^2 + n_{01}{}^2 + n_{10}{}^2 + n_{11}{}^2\right) - \frac{2}{n-1}\left(n_0{}^2 + n_1{}^2\right) - 1$	2	1	5.9915
Poker	$X_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i{}^2\right) - k$	$2^m - 2 = 6$	326	12.5916
Run	$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$	$2k - 2 = 6$	27	12.5916
Correlación	$X_5 = 2\frac{\left(A(d) - \frac{n-d}{2}\right)}{\sqrt{n-d}}$	$N(0,1)$	148.822	1.96

5. CONCLUSIONES

A lo largo de este trabajo se ha profundizado en el estudio de la estructura interna de un criptosistema en el orden de la construcción de un cifrador basado en una sola permutación y su respectivo PRNG, un generador de secuencias seudo-aleatorias, para generar las respectivas claves. Con las dificultades que obedecen al análisis y definición de un conjunto de los posibles ataques realizados por los adversarios.

Even y Mansour [1] establecieron que la construcción de un cifrador mediante una permutación no requería la necesidad de almacenar o generar una multitud de permutaciones y con esta construcción evidentemente no se requirió almacenamiento de permutaciones que confirma dicha afirmación.

Muchas opiniones corroboran que el diseño y análisis de algoritmos convencionales están basados en las propiedades de difusión y confusión. Desde el punto de vista de permutaciones no seleccionadas éstas se construyen usando primitivas realizables. Para la seguridad de los algoritmos, es mejor evaluarlos a la luz de los métodos cripto-analíticos y los principios existentes.

La afirmación de que los PRNG son de diferentes clases de primitivas criptográficas, hay aplicaciones donde el desempeño de PRNG es un problema serio que amerita un nuevo algoritmo. Yarrow-160 en la práctica tiene debilidades en la estimación de la entropía y no en el criptoanálisis. Es necesario continuar probando los mecanismos de la entropía y ésta estará sujeta a futuras investigaciones.

Las reglas de control de re-sembrado son aún un diseño ad-hoc; el estudio extenso podría rendir una mejora en el control de reglas en el proceso re-sembrar.

No está establecido cómo se resiste el compromiso de estado del sistema propuesto. Esto constituye un enorme problema práctico que ha recibido poca atención en la literatura.

BIBLIOGRAFIA

[1]. S. Even and Y. Mansour ,A Construction of a Cipher from a Single Pseudorandom Permutation, Journal of cryptology,Volume 10, Number 3, 1997, Pages 151-162

[2] C.E. SHANNON, Comunication theory of secresy system , Bell system Teech. J, Vol. 28,1949,pp, 656-715.

[3] O. GOLDREICH, S Goldwaser and S. Micali , "How To Construct Random Function " Proceeding of the –25th Annual Symposium of Foundation of computer science, 24-26, 1984.

[4] J. DAEMEN, "Limitation of the even-Mansour Construction " , Katholieke universiteit Leuven Laboratorium Esat , B-3001 heverloe Belgium.

[5] A. MENEZES, P. Van Oorchot , Scott a: castone , " Hand Book Applied Cryptography " , CRC press , 1997

[6] J. KELSEY, B. Schnneir , D. Wagner and C. Hall , "Cryptanalytic Attack on Pseudorandom Number Generators " , fast software Encryption , 5th international Workshop Proceedings Springer-Verlag , 1998 , pp 168-188.

[7] J. KELSEY, B. SCHNNEIR, N FERGUSON, " Yarrow –160 Notes on the design an analysis of the Yarrow Criptographic Pseudorandom Number generator", Counterpane System , 101 E Minnehaha Parkway , Minneapolis , MN 55419, USA. [8] J. VILLALOBOS, "Tipo abstracto de datos en lenguaje c ", Mc Graw Hill , 1995

[9] M. LUBY and C. RACKOFF, "How to construct pseudorandom permutation from pseudorandom function , SIAM Journal computing , 373-386, april 1988.

[10] N. HOLLAND" Stream Cipher and Number Theory " ,North_holland mathematical Library , volume 55 , Elsevier , 1998

[11] SHIMON Even, MANSOUR , Yishay: A Construction of a Cipher From a Single Pseudorandom Permutation, in ASIACRYPT 1991, pages 210-224

A framework for performance evaluation of parallel applications on the Grid

Carlos Figueira

and

Emilio Hernández

and

Eduardo Blanco*
Universidad Simón Bolívar, Departamento de Computación y Tecnología de la Información,
Caracas 1080-A, Venezuela
{figueira,emilio,eduardo}@ldc.usb.ve

Abstract

Performance evaluation of applications running on a Grid is a challenging task. Grid's resources are heterogeneous in nature, often shared, and dynamic, all of which have important implications on the performance of an application executing on the Grid. For instance, applications performance will suffer from perturbation induced by external load on the network or computational nodes. Also, resources allocated to applications may vary between different executions. In this paper, we propose a simple framework that takes into account these factors to allow users to gain knowledge of fundamental performance characteristics of their parallel applications. This framework was incorporated in SUMA, a Grid-enabled platform for the execution of scientific applications in Java. We show some results of the utilization of this framework, which was tested by analyzing and tuning a parallel application.

Keywords: Performance Evaluation, Performance Tuning, Parallel and Distributed Systems, Computational Grids

1 Introduction

The *Grid* or *Computational Grid* [12] has the potential to become the platform of choice for high performance applications. Many of these applications will eventually be either ported to the Grid or developed directly on the Grid. Hence, tools as those used on single system platforms will be adapted to the Grid, and certainly new tools will be developed as well [14, 29].

The Grid gathers distributed resources under a common, simple view. Resources are allocated to applications according to their needs and availability. Resources may include computational nodes, repositories, instruments, etc. They are heterogeneous in nature, often shared, and dynamic, all of which have important implications on the performance of an application executing on the Grid. For instance, they will suffer from perturbation induced by external load on the network or computational nodes. Also, resources allocated to an application may vary between different executions, or even during the same execution.

Performance evaluation on Grids is recognized as a complex problem [15]. Some tools, like Askalon [9], provide a coherent set of tools to help users in the tasks of performance analysis of parallel and distributed applications on the Grid. It follows the architecture proposed by the Global Grid Forum [27] in order to ensure flexibility and interoperability. The tool proposed in [3] gives the user performance information of interactive applications on the Grid.

*This work was partially supported by grants from Fonacit (*Proyecto Agenda Petróleo* 97003588) and from Universidad Simón Bolívar (*Programa de Apoyo a Grupos de Investigación, Decanato de Investigación y Desarrollo*).

However, these tools fail to address important questions arisen in Grid environments, namely the dynamic overhead of the middleware and its impact on the performance, and the performance data scalability. A framework was introduced in [11] to cope with the problem of performance evaluation of applications running on Grids. It proposed the use of *condensed profiles*, which express basic metrics of an application in order to understand its performance and to achieve platform independent optimizations. The condensed profiles included a report of the overhead induced by the Grid middleware. It suggested to normalize computational nodes' power by using benchmarks to characterize platform performance.

In this paper, we extend that work to cope for parallel applications, limiting our scope to Grids whose parallel nodes are accessed through the middleware and that do not vary during an execution. We also explore the use of benchmarks and metrics r_∞, $n_{1/2}$ [20], as alternative means to normalize parallel computational nodes' power.

The rest of the document is structured as follows. In section 2 the extended framework of performance metrics for parallel applications on Grids is briefly described. Section 3 describes, through examples, the performance information handled by the framework, as well as the use of this information to tune a parallel application on the grid environment SUMA [18]. Section 4 introduces two different methods to obtain normalized power and shows how to infer whether platform independent improvements have been achieved. Finally, some conclusions are addressed in section 5.

2 Extended framework

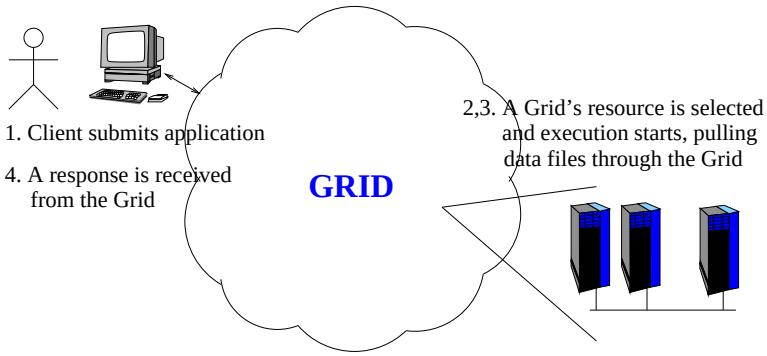


Figure 1: Execution of an application on the Grid

We consider a simplified scenario for the execution of an application on the Grid, as shown in figure 1:

1. A user submits an application to the Grid, specifying a number of execution requirements like, for instance, a parallel platform with MPI.
2. After authentication and access control verification, the Grid's scheduler and resource locator component assigns execution to a given platform.
3. Execution starts on the chosen platform, probably requiring input data files transfers from the user's machine or from a repository (e.g., a remote file server).
4. After execution, a response is produced and returned to the user, which may include application's output files and data, as well as performance information (if requested).

The execution activity will generally consist of more than a single execution. For every execution, the scheduler might choose a different platform, raising the problem of relating performance data from different platforms. Also, environment conditions (e.g., network traffic) may vary between executions. The framework components described below take into account these particular factors.

2.1 Middleware overhead

The middleware, which provides the single system illusion, introduces perturbation of different types:

- *Latency*, regarded as the time elapsed since the application is submitted to the Grid until execution starts at the selected platform. This time encompasses authentication, resource finding, and actual transfer of executables and initial input data.
- *Overhead* caused by remote accesses during execution at the platform, when necessary, including objects, classes and files from the user machine or remote repository.
- *Delay* caused by results transfer to the user.

Information about time and resource usage incurred by perturbations of these three kinds is delivered to the client. It can be used, for instance, to gain accuracy in estimating the actual execution time at the platform. Additionally, Grid administrators can process this information to assess middleware performance.

2.2 Condensed profile

The condensed profile involves performance metrics related to the overall execution and performance metrics of selected nodes of the parallel architecture. In order to achieve scalability, we chose a fixed size performance data format, based on statistical summaries. This fixed size format does not depend on the number of nodes or any other platform characteristic, hence the performance data transfered through the network remain small regardless of the size of the platform. The overall performance metrics include total execution time measured at the front-end and mean and variance of CPU time and communication/synchronization time. These metrics were selected to help in detecting problems related to load balancing, bad granularity or unsuitable data distribution. This information is supported by metrics collected from those nodes that exhibits extreme behavior like, for instance, nodes with highest and lowest execution or communication/synchronization time.

3 Analysis of a parallel execution

We conducted some experiments in order to assess the power of the condensed profiles. These experiments were performed using SUMA, a java-enabled Grid system. [1]

SUMA aims at executing high performance Java byte-code applications [22, 23]. It offers interactive and batch execution on sequential and parallel nodes, transparently allocated by the middleware, as well as added services like profiling [11] and checkpointing [5]. Applications are sent to Execution Nodes, where execution starts, redirecting I/O to client machines, and loading classes and data from client machines. SUMA middleware was originally built on top of commodity software and communication technologies, including Java and CORBA. A recent reimplementation called SUMA/G [4] is partially based on Globus services [26]. In particular, we say that SUMA/Gis a Grid-enabled execution platform because it includes mechanisms for utilizing the Globus Security Infrastructure. By reimplementing some of the SUMA components on top of Globus services we can connect to deployed well-known grids, while keeping SUMA execution model unchanged. Additional functionalities are inherited from Globus, such as the I/O services.

For our experiments, SUMA was deployed on interconnected LAN's located on two buildings of the Universidad Simón Bolivar campus, as shown in figure 2. In such a local deployment, the middleware overhead is low. We focus here on two fundamental questions:

- Does the condensed profile offer a useful picture of the performance of a parallel application?
- Is the condensed profile useful for achieving high level optimizations of a parallel application?

3.1 Experiments environment

Execution Node is taken from a cluster (called CAR), composed by 24 dual 600-800 MHz Pentium III PC's with 512 MBytes RAM, running Linux, interconnected by a 1.9 GHz full-duplex (per link) Myrinet network. The message passing library is LAM-MPI; the JVM is JDK 1.2.2; mpiJava [2] version is 1.1. The back-end profiler is based on MPE [16] and *HProf* [25]. Note the instrumentation overhead is kept low by using statistical sampling (*HProf*) and instrumenting only the actual communication methods (MPE). Condensed profiles are locally stored in each node, then collected and summarized at the front-end processor.

[1] http://www.suma.ldc.usb.ve

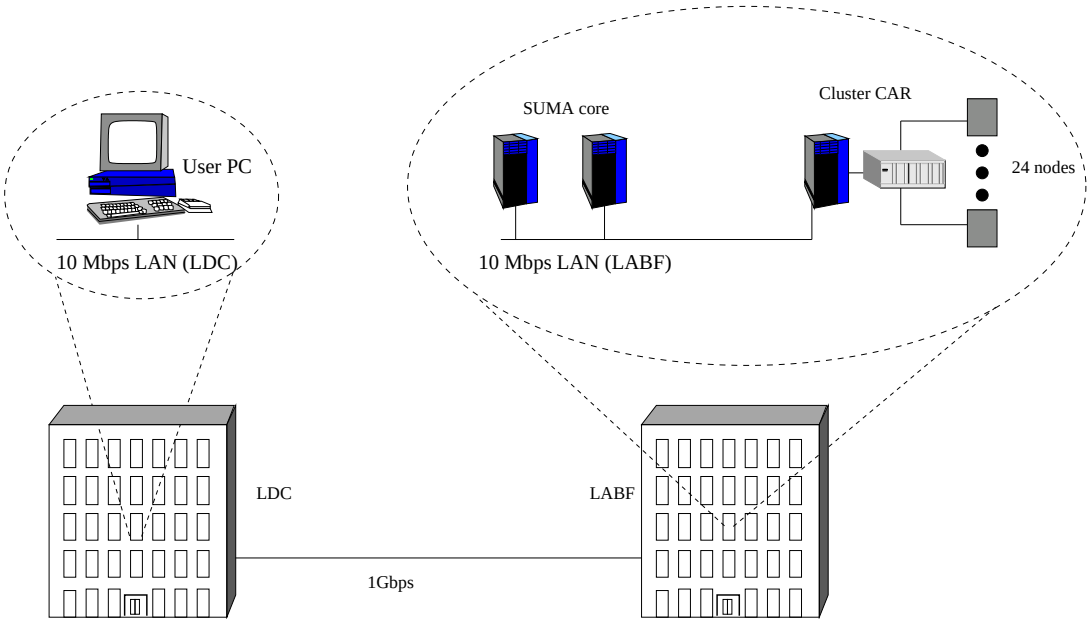


Figure 2: Location of user, SUMA core components and execution resource machines

Core components of the Grid (e.g., Scheduler, CORBA Name Server, etc.) are installed on a dual Pentium III PC with 256 MBytes RAM. The user interface (client) runs on a Pentium IV PC 1.5 GHz with 256 MBytes RAM. All Execution Node, Grid core components and client run on two networks scattered over the campus, as shown on figure 2.

The application is a model of wave propagation in 2D homogeneous surface, whose core computation is a triple loop (2D + time) of medium size (400x400 surface points, computed over 600 time steps). It is a SPMD mpiJava application; the coordinator process initializes and distributes data to the rest of the processes, then computes and receives final results from them. All processes are arranged on a logical linear array; each one of them computes a band of the result matrix, and exchange frontier rows with its neighbors on each iteration of the outer loop. For this experiment, We suppressed saving the results to file.

3.2 Analysis of condensed profile

Total time	88 sec.
Profiling overhead	22 sec.
Middleware overhead	1 sec.

Table 1: Global performance information on 8 nodes, as seen by the user

Time at front-end	64 sec.			
Execution time	Mean:	62 sec.	Variance:	0
Com/Sync time	Mean:	3 sec.	Variance:	48

Table 2: General performance information at the Execution Node

The parallel application is run on 8 nodes of the cluster. Information taken from the condensed profile are shown in tables 1, 2, 3, 4 and 5, where only relevant information is displayed.

Table 1 shows performance information as seen by the user. As expected, the middleware overhead (1 sec. vs 88 sec. of total time) is negligible. Note however that there is considerable overhead (22 sec.) due to profiling (e.g. collecting and processing performance data).

Table 2 shows information recorded at the front-end of the Execution Node. Note the mean communication/synchronization time (3 sec.) is low with respect to mean execution time (62 sec.); it suggests that we might benefit from increasing the number of nodes, reducing the mean execution time at the expense of the communication time. Additionally, the com/sync. time variance (48) suggests that there could be some unbalance among the nodes.

Tables 3, 4 and 5 illustrate an homogeneous behavior across all nodes.

Host name	*car07*
Total Time	62 sec.
CPU Time	59 sec.
Com/Sync Time	3 sec.

Method	**% Time incl.**
`AcousticMPI1.dim2acust`	24
`AcousticMPI1.itera`	24
`AcousticMPI1.main2`	24
`AcousticMPI1.main`	23

Table 3: Node with highest CPU time

Host name	*car0*
Total Time	63 sec.
CPU Time	53 sec.
Com/Sync Time	10 sec.

Method	**% Time incl.**
`AcousticMPI1.main2`	25
`AcousticMPI1.dim2acust`	23
`AcousticMPI1.itera`	23
`AcousticMPI1.main`	21

Table 4: Node with highest communication/synchronization time

Host name	*car01*
Total Time	63 sec.
CPU Time	62 sec.
Com/Sync Time	1 sec.

Method	**% Time incl.**
`AcousticMPI1.main2`	25
`AcousticMPI1.main`	24
`AcousticMPI1.dim2acust`	23
`AcousticMPI1.itera`	23

Table 5: Node with lowest communication/synchronization time

3.3 Execution on several platforms

Additional experiments show how the detailed performance information of most relevant nodes provide useful data about application characteristics when dealing with multiple platforms. Tables 6 and 7 include excerpts from condensed profiles of, respectively, fastest, and maximum communication/synchronization time from executions on clusters with different number of nodes. Tables 8 and 9 complement former table with general performance information about those nodes.

Tables 6 and 7 suggest that the application behaves similarly in all platforms. Note that communication/synchronization time increases as the number of nodes rise from 8 to 16 and 24 nodes; it corresponds with a higher proportion of execution time attributed to `mpi` methods in tables 6 and 7.

3.4 High level tuning

Tables 3, 4 and 5 hint at a set of four methods: `main`, `main2`, `dim2acust`, and `itera`, all belonging to class `AcousticMPI1`, which altogether account for more than 20% of the node execution time and represent by far the most important methods of the application.[2] It turns out that those four methods are part of a calling sequence *main*

[2] Runtime and communication methods account for an important share of the execution time; they are not taken into account for our optimization purposes.

Nodes	Fastest	
	%	Method
4	24	`AcousticMPI1.main2`
	24	`AcousticMPI1.main`
	24	`AcousticMPI1.dim2acust`
	24	`AcousticMPI1.itera`
8	24	`AcousticMPI1.main2`
	23	`AcousticMPI1.main`
	22	`AcousticMPI1.dim2acust`
	22	`AcousticMPI1.itera`
16	24	`AcousticMPI1.main2`
	23	`AcousticMPI1.main`
	22	`AcousticMPI1.dim2acust`
	22	`AcousticMPI1.itera`
	1	`mpi/MPI.InitNative`
24	24	`AcousticMPI1.main2`
	23	`AcousticMPI1.main`
	18	`AcousticMPI1.dim2acust`
	17	`AcousticMPI1.itera`
	6	`mpi/MPI.InitNative`

Table 6: Summary of detailed information of fastest nodes on several platforms (% of inclusive time)

→ *main2* → *dim2acust* → *itera*. We decided to concentrate our optimization effort on `itera`, which contains the core computation loop. Note that while there is communication (i.e., row exchange) inside the outer loop, it is not significant from the point of view of performance, as indicate the communication/synchronization metrics on table 2 (3 sec. vs 64 sec. of total time measured at the front-end).

A tuned version, called `AcousticMPI2`, was generated by modifying the method `itera` to take some independent computations out of the loop. The new general performance is shown on tables 10 and 11. An improvement of 22 % (68 vs 88 seconds) overall and 31 % (44 vs 64 seconds) at the node was achieved.

Playing the role of Grid infrastructure developers and administrators, we decided to investigate why profiling overhead was so high (22 sec.) compared to total time in both original (66 sec.) and tuned (88 sec.) application versions on the 8 nodes cluster. We discovered that one internal performance trace processing was unnecessarily complex. After optimizing the trace processing algorithm, the profiling overhead was reduced from 22 seconds to less than 4 seconds.

4 Normalized power

Performance characterization of a platform aims to express key parameters from which, ideally, it is possible to infer the performance of applications that will run on that platform. There exist different ways of characterizing performance of a platform [17]. We consider here two widespread and simple yet powerful methods: benchmarking and metrics r_∞ and $n_{1/2}$.

4.1 Benchmarking

Benchmarking consists on running one or more programs on the platform, then summarizing results (e.g., execution time, MFLOPS, etc.) in a single number by using, for instance, arithmetic or geometric means [24]. The popularity of this method is due in part to the fact that it naturally takes into account the interaction between the application and the system components (hardware, operating system, libraries, compiler, interpreter, etc.). Normalization is achieved by relating the benchmark execution performance results to a reference platform.

Benchmark accuracy relies on the affinity between the programs used in the benchmark and the applications that will actually run on the platform. Several benchmarks for parallel platforms, in C and FORTRAN, are well established

Nodes	Max. Com/Syn	
	%	Method
4	24	`AcousticMPI1.main2`
	24	`AcousticMPI1.dim2acust`
	24	`AcousticMPI1.itera`
	21	`AcousticMPI1.main`
	3	`AcousticMPI1.exchangeRows`
8	23	`AcousticMPI1.main2`
	22	`AcousticMPI1.dim2acust`
	22	`AcousticMPI1.itera`
	20	`AcousticMPI1.main`
	3	`AcousticMPI1.exchangeRows`
16	23	`AcousticMPI1.main2`
	21	`AcousticMPI1.dim2acust`
	21	`AcousticMPI1.itera`
	16	`AcousticMPI1.main`
	7	`mpi/Comm.Recv`
24	24	`AcousticMPI1.main2`
	17	`AcousticMPI1.main`
	16	`AcousticMPI1.dim2acust`
	16	`AcousticMPI1.itera`
	6	`mpi/Comm.Recv`

Table 7: Summary of detailed information of nodes with maximum communication/synchronization time on several platforms (% of inclusive time)

Nodes	fastest	
	T. CPU	T. Com/Syn
4	118	2
8	63	2
16	42	2
24	39	3

Table 8: Summary of general performance information of fastest nodes (time in seconds)

and are commonly used by the high performance computing community [1, 7]. Only recently, some benchmarks have been ported or developed in sequential and parallel Java versions [13, 8].

4.2 Performance characterization using r_∞ and $n_{1/2}$

A model based on metrics r_∞ and $n_{1/2}$ has been used to characterize performance of parallel architectures [20]. While conceived for vector architectures, the model has been extended to other types of architectures, including distributed memory parallel machines [19]. It describes execution time using a first-order approximation shown in equation (1), where n is number of operations (typically, floating point operations); r_∞ is the peak performance, often expressed in MFLOPS; $n_{1/2}$ is the number of operations to reach half the peak performance.

$$t = \frac{n + n_{1/2}}{r_\infty} \tag{1}$$

4.3 Quality of normalized power

We conducted experiments to assess two alternative methods to obtain the normalized power:

1. Benchmarking. The normalized power is the ratio of execution time of the benchmark on a given platform to the execution time on a reference platform.

Nodes	**Max. Com/Syn**	
	T. CPU	T. Com/Syn
4	105	16
8	58	8
16	32	14
24	33	13

Table 9: Summary of general performance information of nodes with maximum communication/synchronization time (in seconds)

Total time	68 sec.
Profiling overhead	22 sec.
Middleware overhead	1 sec.

Table 10: Global performance information of tuned version on 8 nodes

2. Ratio of r_∞. The normalized power is the ratio of platform's r_∞ to the reference platform's. It was possible to simplify and discard $n_{1/2}$ because experiments showed that accuracy did not vary significantly [10].

The experiments consist on running an application on the reference platform, then on a test platform. The execution time measured on the test platform is compared with a *predicted time*, obtained by applying equation (2)

$$\textit{Predicted time} = \frac{\textit{Time on reference platform}}{\textit{Test platform's normalized power}} \tag{2}$$

In order to be able to compare measured times, middleware overhead must be subtracted from the time measured at the node.

The application to be used as benchmark is a reduced version of `AcousticMPI1`. The reference platform is a two node cluster with 10 Mbps Ethernet, each node consisting of a dual processor 600 MHz Pentium III, with 256 MBytes SDRAM PC133 (512 cache *off-chip*), and Linux RedHat 6.2.

Table 12 shows prediction results for different cluster sizes using normalized power based on benchmarks. Table 13 shows the prediction results using r_∞ for different cluster sizes.

As indicated in tables 12 and 13, both methods have high (except for 24 nodes) and similar prediction quality. This is related to a good match between the benchmark used for computing the normalized power, and the actual application used in our experiments. In the general case, as Grids will accommodate a possibly high diversity of applications, the quality will be lower than in our experiments.

High error for 24 nodes (31.4 %) exposes a deficiency, related to sensibility to the characteristics of the application used to estimate platforms' performance, both for benchmark and r_∞, $n_{1/2}$ based methods. In this particular case, the application behaved well for smaller clusters but, when used to characterize a 24 nodes cluster, it turned out to be too small to exploit (and hence capture) the platform power.

4.4 Platform independent tuning

Normalized power can be used to compare performance on heterogeneous platforms. Given the improvement shown in section 3.4 by version `AcousticMPI2` measured on an 8 nodes cluster, we can estimate whether there will be improvement in other platforms using normalized power.

Table 14 presents the estimated execution time and speedup of tuned code `AcousticMPI2` (version 2) with respect to original code `AcousticMPI1` (version 1) on two different platforms:

Time at front-end	44 sec.			
Execution time	Mean:	42 sec.	Variance:	0
Com/Sync time	Mean:	2 sec.	Variance:	14

Table 11: General performance information of tuned version at the Execution Node

Nodes	Normal. Power	Time	Predicted	Error
2	1.5	236	231	**1.8**
4	2.9	123	117	**4.7**
8	5.5	68	61	**9.5**
12	6.1	58	55	**4.0**
16	8.3	48	41	**14.5**
24	10.6	47	32	**31.4**

Table 12: Predicted time (seconds) and error (%) using benchmarks for normalized power

Nodes	Normal. Power	Time	Predicted	Error
2	1.4	236	236	**0.1**
4	2.7	123	120	**2.1**
8	5.3	68	62	**8.7**
12	5.5	58	55	**2.3**
16	7.7	48	43	**10.7**
24	10.2	47	32	**31.4**

Table 13: Predicted time (seconds) and error (%) using r_∞ for normalized power

- Two nodes from cluster CAR, described in section 3.1.
- The reference platform, a two nodes cluster described in section 4.3.

Measurements on 8 nodes are also included for comparison purposes. The estimated values (in italics) for tuned version (**`Ver. 2`**) on each platform (2 nodes of cluster CAR and 2 nodes of reference platform) are obtained by using the execution time of that version on the 8 nodes platform, and the ratio of the normalized powers (using r_∞). The values for the original version (**`Ver. 1`**) were actually measured. The speedup shown compares the estimated execution time of the tuned version with respect to the measured execution time of the original version.

	Time (sec.)		
Platform	**`Ver. 1`**	**`Ver. 2`**	**Speedup** (%)
8 nodes	64	44	31
2 nodes	236	*167*	*29*
2 nodes (ref.)	329	*233*	*29*

Table 14: Estimated execution time and speedup of tuned application on two platforms (*estimated values in italics*)

Results shown on table 14 indicate that we could expect the new version to run faster than the original version on relatively different platforms.

5 Conclusions

The performance evaluation of applications on the Grid must take into account distinctive features such as heterogeneity, dynamics and middleware overhead.

The framework proposed in this work captures simple yet essential metrics to understand performance from a high point of view. It separates the influence of the middleware from the application performance, and reports high level and scalable metrics of what actually happens in the execution platform. The middleware overhead is reported thanks to monitoring mechanisms built in the middleware components. Scalability is achieved through simple statistical reduction techniques locally processed, such as to avoid the communication of large sets of performance data.

Our experiments demonstrated that the framework can be effective in identifying coarse performance characteristics of an application, leading to high level, platform independent optimizations. Also, overhead metrics built in the system allowed us to detect a Grid infrastructure implementation problem which incurred in considerable overhead.

Execution on heterogeneous platforms needs a basis to compare performance data (see [28]). We showed how two different method, benchmarking (as in Netsolve [6]) and r_∞, can be used to determine a normalized power, which can in turn effectively serve for that purpose. Incorporating the recently proposed benchmarks in parallel Java in our framework will help in improving the normalized power accuracy.

Our current approach exhibits some limitations that motivate future work. For instance, using dynamic [21] instead of postmortem instrumentation, would enable the performance evaluation of long running applications. Hence, performance data of partial execution would be periodically transmitted during the application execution.

Further experiments, using different classes of applications running on widely deployed Grids, will be carried out to better assess the framework. We also plan to add facilities to process condensed profiles in order to automatically report potential performance problems to users.

References

[1] D. H. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The NAS Parallel Benchmarks. Technical Report RNR-94-007, NASA Ames Research Center, March 1994.

[2] M. Baker, B. Carpenter, G. Fox, and Sung Hoon Koo. mpiJava: An Object-Oriented Java Interface to MPI. *Lecture Notes in Computer Science*, 1586, 1999.

[3] Bartosz Balis, Marian Bubak, Wlodzimierz Funika, Roland Wismller, Marcin Radecki, Tomasz Szepieniec, Tomasz Arodz, and Marcin Kurdziel. Performance Evaluation and Monitoring of Interactive Grid Applications. *Lecture Notes in Computer Science*, 3241:345–352, 2004. Recent Advances in Parallel Virtual Machine and Message Passing Interface: 11th European PVM/MPI Users Group Meeting.

[4] Y. Cardinale and E. Hernández. Parallel Checkpointing on a Grid-enabled Java Platform. *Lecture Notes in Computer Science*, (European Grid Conference EGC2005), February 2005. To appear.

[5] Yudith Cardinale and Emilio Hernández. Checkpointing Facility in a Metasystem. *Lecture Notes in Computer Science*, 2150:75–79, August 2001. 7th International Euro-Par Conference.

[6] Henri Casanova and Jack Dongarra. Netsolve: A Network Server for Solving Computational Science Problems. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(3):212–223, Fall 1997.

[7] J. Dongarra and T. Hey. The PARKBENCH Benchmark Collection. *Supercomputer*, 11(2-3):94–114, 1995.

[8] EPCC. The Java Grande Forum Benchmark Suite. http://www.epcc.ed.ac.uk/computing/-research_activities/java_grande/index_1.html, 2005.

[9] Thomas Fahringer, Alexandre Jugravu, Sabri Pllana, Radu Prodan, Clovis Seragiotto Jr, and Hong-Linh Truong. ASKALON: a tool set for cluster and grid computing. *Concurrency and Computation: Practice and Experience*, 17(2-4):143–169, February - April 2005.

[10] Carlos Figueira. *Evaluación de Desempeño de Aplicaciones en Metasistemas*. PhD thesis, Universidad Simón Bolívar, 2002.

[11] Carlos Figueira and Emilio Hernández. Profiling Facility on a Metasystem. *Lecture Notes in Computer Science*, 2110:42–51, 2001. Conference on High Performance Computer and Networking (HPCN'01).

[12] Ian Foster and Carl Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*, chapter Computational Grids, pages 15–51. Morgan Kaufmann Publishers, Inc., 1999.

[13] Michael Frumkin, Matthew Schultz, Haoqiang Jin, and Jerry Yan. Implementation of the nas parallel benchmarks in java. Technical report, Nasa Advanced Supercomputing, 2002.

[14] Global Grid Forum. Grid Performance and Information Services, 2003. http://www.gridforum.org/.

[15] G. Gombas, Z. Balaton, and Z. Nemeth. Performance evaluation on grids: Directions, issues and open problems. In *12th Euromicro Conference on Parallel, Distributed and Network based Processing*, 2004.

[16] William Gropp and Ewing Lusk. User's guide for mpe: Extensions for mpi programs, electronic publication. http://www-unix.mcs.anl.gov/mpi/mpich/docs/mpeguide/paper.htm.

[17] Emilio Hernández. *A Methodology for the Design of Parallel Benchmarks*. PhD thesis, University of Southampton, 1996.

[18] Emilio Hernández, Yudith Cardinale, Carlos Figueira, and Alejandro Teruel. SUMA: A Scientific Metacomputer. In *Parallel Computing: Fundamentals and Applications. Proceedings of the International Conference ParCo99*, pages 566–573. Imperial College Press, 2000. ISBN 1-86094-235-0.

[19] R. Hockney. Performance parameters and benchmarking of supercomputers. *Parallel Computing*, 11(2-3):94–114, 1995.

[20] R. Hockney and C. R. Jesshope. *Parallel Computers: Architectures, Programming and Algorithms*. Adam Hilger Ltd., Bristol, UK, 1981.

[21] Jeffrey K. Hollingsworth, Barton P. Miller, and Jon Cargille. Dynamic program instrumentation for scalable performance tools. In *Proceedings of the 1994 Scalable High-Performance Computing Conference*, pages 841–850, 1994.

[22] Java Grande Forum. http://www.javagrande.org.

[23] S. Markidis, G. Lapenta, W. B. VanderHeyden, and Z. Budimlic. Implementation and performance of a particle-in-cell code written in Java. *Concurrency and Computation: Practice and Experience*, 17:821–837, 2005.

[24] David Patterson and John Hennessy. *Computer Architecture: A Quantitative Approach*. Morgan/Kaufmann, second edition, 1996.

[25] Sun Microsystems. Javatm virtual machine profiler interface (jvmpi). http://java.sun.com/j2se/1.4.2/docs/guide/-jvmpi/jvmpi.html, Consulted April 2005.

[26] The Globus Alliance. The Globus Toolkit. http://www.globus.org/.

[27] B. Tierney, R. Aydt, D. Gunter, W. Smith, M . Swany, V. Taylor, and R. Wolski. A grid monitoring architecture. Global Grid Forum. http://www-didc.lbl.gov/GGF-PERF/GMA-WG/.

[28] Jeff Vetter and Daniel Reed. Managing performance analysis with dynamic statistical projection pursuit. In *SC'99*, 1999.

[29] VGrADS. The Virtual Grid Application Development Software Project. http://vgrads.rice.edu.

An Electronic Marketplace: Agent-based Coordination Models for Online Auctions

Ayşe Morali
Technische Universität Darmstadt; Darmstadt, Germany

Leonardo Varela
Universidad de Los Andes; Bogotá, Colombia

Carlos A. Varela
Rensselaer Polytechnic Institute; Troy, NY 12180, USA

morali@winf.tu-darmstadt.de, l-varela@uniandes.edu.co, cvarela@cs.rpi.edu

Abstract

Different coordination models exist for managing concurrency in complex distributed systems, for example, direct interaction, tuple spaces, hierarchical structures, and publish-and-subscribe mechanisms. Online auctions are complex distributed systems, where the choice of coordination model can significantly impact the performance of multiple software agents, acting on behalf of human buyers and sellers. In this paper, we evaluate analytically and experimentally different auction types and coordination models in an *electronic marketplace*. Three important metrics are: the number of software agents, the number of messages exchanged between these agents, and the number of migrations performed by the agents. We conclude that dynamically choosing the number of agents and the coordination model can improve quality of service in electronic commerce applications. Furthermore, in online auctions where there is a lot of interaction between software agents, migration of agents to virtual stores and local bidding is more fair and efficient than using remote interaction. This observation has the potential to significantly improve and redefine existing online auction systems such as eBay.

Keywords: Coordination models, multi-agent systems, electronic commerce, online auctions

1 INTRODUCTION

The World-Wide Web [4] has fueled a plethora of electronic services that have either matched or surpassed previously existing services since their inception only a decade ago. For example, electronic newspapers today are widely read, conceivably as much as or even more than physical newspapers. They reach faster a larger worldwide audience. They introduce significant and challenging new problems, for example: on-screen formatting and navigation, advertisement methodologies, back issue indexing and search procedures, and online interaction fora such as blogs. Ultimately, electronic newspapers are an invaluable complement to their (much older) physical newspaper counterparts, in terms of bridging the gap between information provision and use.

Electronic commerce has also seen just the tip of the iceberg in terms of potential services to customers and companies alike. Only a few years ago, very few people dared enter their credit card information in an online web form. Today, it is common to use the web as an electronic purchasing medium, for example, to buy airline tickets on specialized electronic travel agencies on the web, to offer company stocks on trading sites, or to participate in online auctions, just to name a few.

As the sophistication of web services grows, as information becomes both human and machine-readable [9], and as multi-agent systems become pervasive in the electronic information age, it becomes imperative to re-think traditional processes in understanding how to tackle the significant and challenging new problems introduced by the new electronic commerce medium.

In this paper, we consider a futuristic electronic marketplace consisting of software agents representing human buyers, sellers, and auctioneers. We study and analyze different types of auctions in terms of their potential to satisfy users, and in terms of their electronic complexity, that is, number of software agents, cost and patterns of communication, and likelihood of inducing commercial transactions. We also study and analyze different multi-agent coordination models in terms of computational complexity, and electronic commerce metrics, such as user satisfaction. Finally, we envision a system that requires minimal human intervention and that dynamically adapts the electronic commerce strategy from the perspective of the sellers and auctioneers (e.g., type of auction) and from the perspective of the buyers (e.g., the inter-agent coordination model) to optimize electronic customer satisfaction. Ultimately, we aim to create an adaptive electronic marketplace that can successfully complement today's commercial activities and bring in a new perspective to the fair and efficient exchange of goods.

Structure of Paper Section 2 surveys existing auction types and coordination models. Section 3 introduces the electronic marketplace architecture and the dynamic coordination model selection approach. Section 4 presents analytical models for different auction types and coordination models in terms of metrics such as number of agents, messages and migrations. This section also shows experimental results obtained through a software prototype. Section 5 concludes with a discussion of contributions and potential future work.

2 RELATED WORK

2.1 Auction Types

Online auctions mimic auctions in the physical world and differ according to many parameters like the role of the sellers or buyers and the sort of pricing or bargaining. In this section, we introduce different types of auctions as they might be used in the cyber world.

Software agents participate in an auction on behalf of their owner, who can be a seller or a buyer. Considering the roles of the agents, auctions are classified as *seller auctions* and *buyer auctions*. Seller auctions are the classical form of auctions. Bidders place offers on one or more items. On the other hand, buyer auctions, which are also known as *reverse auctions*, work the other way around. Sellers compete with each other to satisfy the buyer. There are also *double auctions*, which are a combination of buyer and seller auctions.

Depending on the pricing and the bargaining schemas, auctions are categorized into *iterative auctions* and *sealed-bid auctions*. *English auctions* and *Dutch auctions* belong to the first group, where the price ascends iteratively. *First-price-sealed-bid auctions* and *Vickrey auctions* are the major examples of auctions in the second group. Following, we are going to describe these different types in more detail.

English auction: This is the most frequently used and common method and is known also as the *open-outcry auction* or the *ascending-price auction* [13]. The auctioneer begins here with the lowest acceptable price, which is not secret, and proceeds to solicit successively higher bids from the buyer agents, and ends with a timeout. The item is sold to the highest bidder agent at the price of her last bid [11]. In an English auction, the buyer agents may bid several times or react to the bids of others arbitrarily. The information flow of English auctions can be seen in the sequence diagram in Figure 1a.

One variation of the English auction is the *open-exit auction* [15]. Here the prices rise continuously, and bidders must publicly announce that they are dropping out when the price is too high. Once a bidder has dropped out, she may not reenter. In another variation, an auctioneer calls out each asking price and bidders lift a paddle in the online version, send an accept message, to indicate a willingness to pay that amount. Then the auctioneer calls out another price, etc. *Yankee auctions* [19] are another type of English auction, where multiple items of the same kind are negotiated in one auction. The items are allocated after the auction is finished, depending on different criteria like desired amount of items and maximum budget of the agent.

Dutch auction: This auction is also known as the *descending-price auction* [14]. It has an iterative format. Bidding starts at a relatively high price and is progressively lowered [24]. The agent that accepts the lowered price gets the item. All of the buyers have the same chance to get the item, but the ones that hesitate too long miss it. When the goods are exhausted, the bidding is over.

The message flow of the Dutch auctions are very similar to the message flow in English auctions. The only difference is that it does not end with time over but with price acceptance of a buyer agent. In order

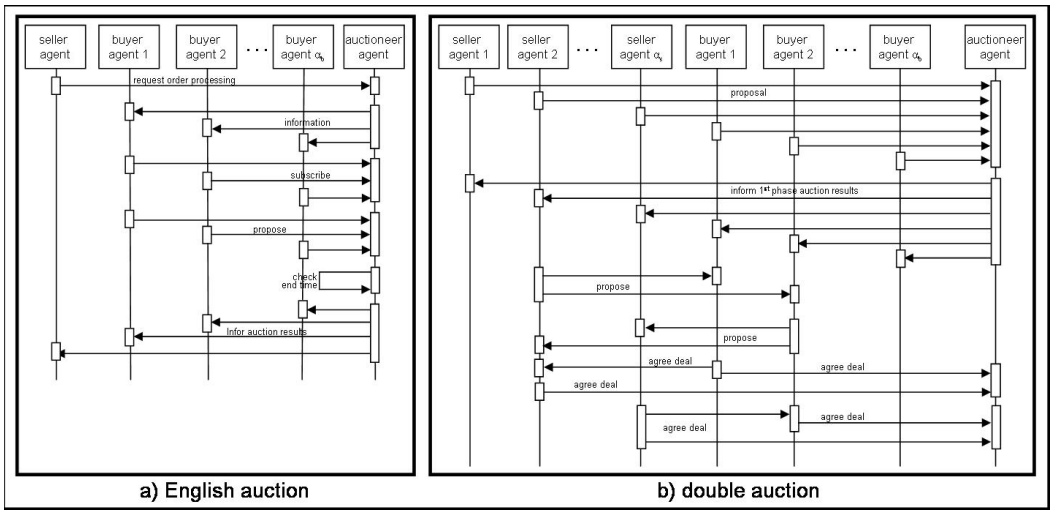


Figure 1: Sequence diagram for English and double auctions

to protect the seller from dis-profit a lower bound can be optionally predefined.

When multiple units are auctioned, normally more buyers are present. The first winner takes one of the items and pays her price and later winners pay less [24]. Once again, when the goods are exhausted, the bidding is over.

It is believed that the English system may be inferior to the Dutch in one area. From the seller's point of view, the key to any successful auction is the effect of competition on the potential buyers, and in an English auction, the under-bidder usually forces the bid up by one small step [14]. The winner may end up paying well according to her point of view but the seller does still not receive the maximum price.

Reverse auction: In reverse auctions, the successful bidder is determined by the lowest price submitted to the auctioneer at the conclusion of the auction [12]. While potential suppliers are underbidding each other, the buyer does not do anything but only observes the negotiation.

Reverse auctions have the following flow [10]: The buyer submits a list of items to the auctioneer. The auctioneer invites sellers to participate, with a specified start and closing time. All bidder's identities are kept confidential during the auction. Bidders submit prices that are ranked and disseminated to all bidders as new bids are made. An extension of the auction closing time may be triggered for a pre-determined period of time if one of the top bid rankings changes. The auction closes once no new bids are placed and the original or extended closing time expires. All bidders are immediately notified of the final bid rankings, and the auctioneer notifies the buyer of the bidding results.

Sealed-bid auction: In sealed-bid auctions, each bidder submits one bid without knowledge of all other bids [24]. That is what being sealed denotes, as opposite to open-outcry like the English or Dutch varieties. Each agent can bid only once and the other agents do not know the buying strategy or the budget of their competitors. In *first-price-sealed-bid auctions* [16] after all bids are executed they are co-instantaneously declared and the highest bidder gets the item for the price she bid. That is where the term "first-price" comes from. In the case of a buyer auction the supplier with the lowest bargain sells the item.

When multiple units are being auctioned, it is called *discriminatory auction*. In that case, the bids are sorted from high to low and the units are sold to the bidders, starting with the highest, until the supply is exhausted. Each bidder has to pay the amount she bid. This results in different prices for different winners. This is where the term "discriminatory" comes from [16].

Vickrey auction: Like the first-price sealed bid auction, there is also a *uniform second-price auction*, known with its finder's name: Vickrey [25]. This auction works also with sealed bids [24]. The difference is that in the Vickrey auction the bidder who made the highest bid gets the good at the price of the highest non-winning bid. If multiple units are being auctioned, all winners pay the price of the highest non-winning bid.

Double auction: Double auctions have been the principal trading format in U.S. financial institutions for over a hundred years [8]. In double auctions, as can be seen from the sequence diagram in Figure 1b,

both sellers and buyers submit bids which are then ranked highest to lowest to generate demand and supply profiles [23]. From the profiles, the maximum quantity exchanged can be determined by matching the selling offers (starting with lowest price and moving up) with demand bids (starting with highest price and moving down) [7]. This format allows both buyers and sellers to negotiate at any particular moment.

One interesting variation of double auction is the *Double-Dutch auction* [8], where a buyer price clock starts at a very high price and goes downward. At some point the buyer stops the clock and bids on the unit at a price complimentary to her. At this point a seller clock starts upward from a very low price and continues upwards until a seller stops it and offers a unit at that price. Then the buyer clock resumes once again downwards, etc. When the two prices cross, the auction is over. The double auction has many variants and is evolving rapidly. Economists believe that the double auction will have many applications as auctions become computerized [17].

2.2 Coordination Models

Actors: The actor model of computation [1] provides a good starting point for building multi-agent systems because of its asynchronous nature of communication, distributed memory, and dynamic reconfiguration properties. Actors coordinate activities by exchanging messages sent asynchronously. In response to a message, an actor may: (1) change its otherwise completely encapsulated state, (2) send messages to other known actors, (3) create new actors, or (4) migrate to a new location.

Due to asynchronous communication and state encapsulation, actor migration is much simpler than object migration, since no execution stacks are present due to nested method invocations and no more than one logical thread is ever accessing the internal state of an actor. As long as a universal naming scheme exists to refer to mobile actors, communication between actors is location transparent [20].

While asynchronous message passing is a sufficient primitive for reasoning about open distributed systems (see e.g., [2]), higher-level coordination mechanisms are needed to effectively build complex multi-agent systems. Actor programming languages, such as SALSA [22], provide additional coordination constructs built over message passing, that facilitate building complex communication protocols, without breaking the dynamic reconfiguration aspects of the model. Coordination constructs include futures, token-passing continuations, join blocks, and first-class continuations.

Tuple Spaces: Linda [6] introduced a powerful coordination paradigm consisting of processes communicating through a shared tuple space. Processes may: (1) write a tuple to the space, (2) read a tuple from the space leaving it in the space, or (3) read and remove a tuple from the space.

An advantage of the tuple space model is that it enables coordination to trascend space and time boundaries since: (*a*) processes can be in different computers when communicating with a tuple space, and (*b*) a tuple can outlive its writing process and therefore it can be read by a process that did not exist at the time the tuple was created.

While tuple spaces go a long way in providing a natural shared memory abstraction for inter-process coordination, their main limitation lies in the difficulty of scaling up coordination to a very large number of processes due to the bottleneck introduced at the tuple space. The logical centralization of a tuple space also imposes a significant overhead in terms of remote message passing in an open distributed system.

Publish and Subscribe: Publish and subscribe models have been proposed in the literature (e.g., [3, 5]) to enable what can be regarded as *active* tuple spaces. Rather than expecting processes to continuously poll for a given tuple in a tuple space, publish and subscribe strategies enable processes to *subscribe* to specific patterns of messages or tuples, and to *publish* tuples with the expectation that all currently subscribed processes will receive a copy of the tuple.

An advantage of publish and subscribe is that multiple subscribers which are co-located can be serviced more efficiently by creating an overlay tree structure and *multicasting* (rather than broadcasting) new information as it becomes available. For example, if web customers are watching the Soccer World Cup 2006 originating in Germany, the video stream packets need only travel once from Europe to the American continent, then be forwarded to intermediate nodes representing North and South America, and so on, which is obviously much more efficient than *broadcasting* the video streams, which would imply many one-to-one connections from America to Europe.

Several major complexities arise when considering a publish and subscribe mechanism including how to express subscription constraints, and how to dynamically maintain an efficient multicasting overlay network, in the presence of multiple information sources and stringent reliability requirements.

Hierarchical Coordination: The hierarchical model for coordination of concurrent activities [21] advocates the creation of groups of actors, or *casts*, to serve as units of coordination among related processes. An actor group is coordinated by a *director*, a special actor in charge of intra-cast coordination. Directors may themselves belong to coordination casts, thereby creating a coordination hierarchy.

The hierarchical model is based on the observation that intra-cast communication is orders of magnitude more frequent that inter-cast communication. This so-called *near-decomposability* property of hierarchic systems [18] allows for their natural evolution and scalability. Simon illustrates this argument with examples drawn from systems such as multi-cellular organisms, social organizations, and astronomical systems.

In the context of electronic commerce, multiple software agents representing a buyer form natural hierarchies with directors coordinating purchases of specific items and item categories. The highest-level director within a group of software agents representing a buyer may enforce user constraints such as the maximum budget to use on acquiring a specific shopping list.

The main advantage of hierarchical coordination over other models is its scalability. The major drawback is that in simple electronic commerce scenarios, it can induce more messaging overhead than it would be necessary with a flat coordination structure.

3 APPROACH: AN ELECTRONIC MARKETPLACE

In order to evaluate different coordination models and auction types, we have created an online marketplace, where many buyer and seller agents come together and negotiate. In this section, we describe the architecture of this marketplace, and the criteria we used to indirectly measure user satisfaction.

The electronic marketplace is a system that enables dynamically choosing the number of agents, agent migrations, and the number of messages sent, for a given buyer. The goal is to increase the possibility of the agents to get all the products desired by the buyer within the given budget, and to accomplish this task in the given time period. Furthermore, from the perspective of the sellers and auctioneers, the system enables to dynamically choose the most suitable auction type, with the goal of optimizing their satisfaction.

The marketplace consists of four main components (see Figure 2): Coordination Model Manager (CMM), Seller Manager (SM), Stores and the Transaction Manager (TM). CMM is an interface between the buyer agents and the stores. It is responsible for choosing the most suitable coordination model by analyzing the items each buyer wants to buy and the items sold in different stores. SM is the interface between the seller agents and the stores. It is responsible for associating seller agents to stores. Stores model physical stores in the real world. They are in charge of spawning auctions, where agents negotiate for selling or buying items. In each store there are many auctions going on simultaneously. The seller and buyer agents come together in auctions. Every auction is managed by an auctioneer agent, which is responsible for deciding the type of auction and reporting the transactions to the TM. TM is the storing unit of the marketplace. All of the transaction results flow to its database.

When a customer enters the marketplace, it gives the items in its item list to the CMM. CMM chooses the coordination model and according to the model creates and sends agents to stores. At the same time, sellers of the real world, represented by seller agents, enter the marketplace and give the specifications of their items to the SM. SM contacts the stores, and either sends a seller agent for each new item to the store or creates a new store for this seller.

The main criterion that auctioneers take into consideration is the number of buyer and seller agents that supplies or demands an item. Since the seller/buyer wants to sell/buy the items in a predefined time period, time is the next important criterion. CMM decides on the coordination model, creates buyer agents and their itinerary, and sends them to stores. The number of messages sent and the number of agent migrations, is strongly affected by the coordination model, and strongly affects user satisfaction.

The indicator of success for our approach is user satisfaction. Satisfaction is measured for buyers by the percentage of items bought and the average price paid for each item as compared to the market price—the average price paid for that item by all the agents in the system. For the seller, user satisfaction is measured by the percentage of the items sold and total revenue. Finally, for the auctioneers, satisfaction is measured by the number of transactions.

The agents representing the same sellers/buyers are responsible for buying/selling all of the items in their item lists, in a limited time, and with a limited budget. If there are communication delays or failures, agents may end up buying more items than desired or getting back with empty hands. The number of remote messages sent between the agents, which are bidding at the same time in different auctions, are more critical

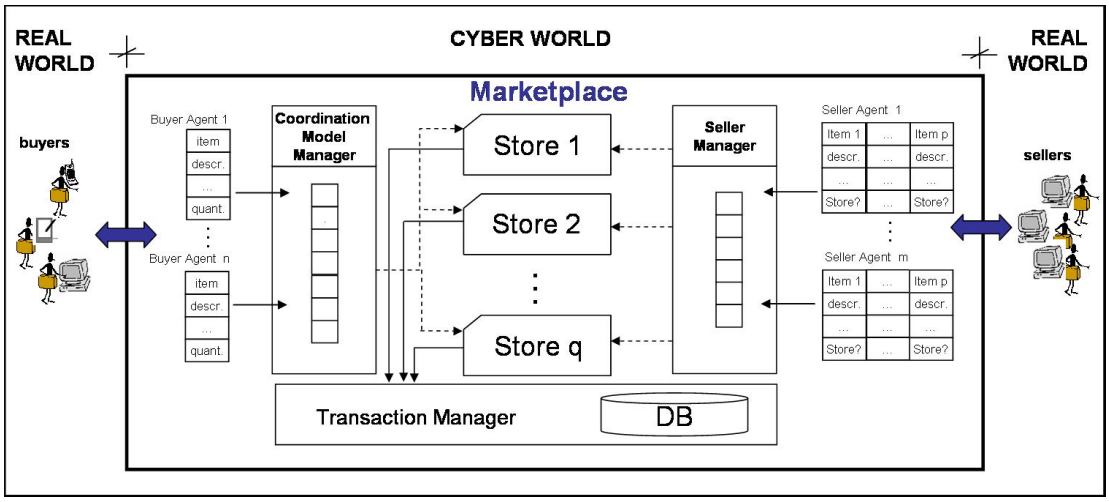


Figure 2: Architecture of the marketplace

than the internal messages, because they require more time. The number of agent migrations depends also very strongly on the chosen coordination model. Since the number of messages sent between the agents increases proportionally to the number of agents, the coordination model should help the agent owner to buy or sell the items with a minimal number of agents.

4 ANALYSIS AND SIMULATION RESULTS

In this section, we are first going to concentrate on the analytical models for the major auction types and coordination models. Later, we will move on to the simulations and results of our electronic marketplace prototype.

The analysis of the auction type and the coordination model are orthogonal to each other. Considering different types of auctions, there are two main variables affecting the success of an auction system. These are number of agents and number of messages. The coordination model affects the success of an auction environment through three variables. These are the number of agents, the number of messages and the number of migrations. The number of agents and number of messages affect each other directly. Since agents communicate with each other through messages, the number of messages increases geometrically with the increasing number of agents.

In an auction environment, time plays a very important role. Being able to migrate where the auction is taking place makes the bidding process less expensive and more fair. In an actor-based auction system the latency and bandwidth differences become less relevant than in a remote-bidding auction system like eBay. In remote-bidding auction systems, users complain about fairness of the bidding process, mostly in the last minute bid. The significance of migration versus remote communication will be detailed in the simulation section.

4.1 Comparison of the Auction Types

In this subsection we are going to compare the number of messages and the number of agents for the most significant auction types. This will help us decide under which conditions which auction type is the most efficient.

The variables of our mathematical approach are as following: α_b, symbolizes the number of buyer agents, α_s, the number of seller agents, β the average bids per agent that take place until an item is sold in an auction, σ, is the number of miniauctions in a double auction, κ, the number of messages, and α, the total number of agents.

The total number of agents is obtained in every auction type by adding the number of buyer, seller, and

	English	Reverse	Sealed-bid	Double
α_b	α_b	1	α_b	α_b
α_s	1	α_s	1	α_s
α	$\alpha_b + 2$	$\alpha_s + 2$	$\alpha_b + 2$	$\alpha_b + \alpha_s + 1$
κ	$\alpha_b(3+\beta)+2$	$\alpha_s(3+\beta)+2$	$4\alpha_b + 2$	$2(\alpha_b + \alpha_s) + \sigma(\beta + 1)$

Table 1: Number of agents and messages in different auction types.

auctioneer agents. Since there is only one auctioneer in every auction type, the general formula is as follows:

$$\alpha \ = \ \alpha_b + \alpha_s + 1$$

There are five main stages in an auction process: The creation, notification, subscription, bidding process and notification of results of the auction. Referring to these stages, the computation of the number of messages needed to complete an English auction in our model is as follows: In the first stage, the seller agent requests the auctioneer to create a new auction by sending one message. In the second stage, the buyer agents request to join the auction. Such process generates as many messages as the number of buyer agents requesting. In the third stage, the auctioneer accepts or refuses the request of the buyer agents. The same number of messages as in the second stage are generated. In the fourth stage, one buyer agent places a bid by sending one message. The auctioneer notifies the other participants of the new bid. The total number of messages generated in this stage is equal to the total number of buyer agents per bid. In the final stage, the auctioneer notifies the seller and the buyers of the auction result. The total number of messages generated in this stage is equal to the number of buyer agents plus one. By adding the number of messages in these five stages, we obtain the formula in the Table 1 for the total number of messages:

$$\kappa \ = \ 1 + \alpha_b + \alpha_b + \beta\alpha_b + (\alpha_b + 1) \ = \ 1 + 3\alpha_b + \beta\alpha_b + 1 \ = \ 1 + \alpha_b(3+\beta) + 1 \ = \ \alpha_b(3+\beta) + 2$$

The reverse auction is similar to the English auction, but the sellers and the buyers change their roles. Consequently, the number of messages remains the same.

In sealed-bid auctions the total number of messages are deducted from the English auction. This is because there is only one bid per agent in the fourth stage, and the rest of the stages remain the same as the English auction.

In the first stage of a double auction, both buyers and sellers send their bids. That process generates $(\alpha_b + \alpha_s)$ messages. In the second stage, the auctioneer notifies the matching sellers and buyers that they are welcome to the second round, where miniauctions between them take place. It also notifies the not-matching buyers or sellers of their elimination from the auction. This notification stage generates $(\alpha_b + \alpha_s)$ messages. Therefore, in the first round in total $2(\alpha_b + \alpha_s)$ messages are generated. In the second and consequent round of a double auction, the number of seller and buyer agents is the same, and equals $min(\alpha_b, \alpha_s)$. In this round, σ miniauctions are situated, where $\sigma = min(\alpha_b, \alpha_s)$. In every miniauction, $(\beta + 1)$ messages on average are communicated, caused by biding done and the final notification to the auctioneer. Therefore, in the second round of a double auction, $\sigma(\beta + 1)$ messages are generated. By adding the number of messages in these two rounds, we obtain the formula for the total number of messages in Table 1.

If there is one buyer and several sellers then the auctioneer generates a reverse auction. In the opposite case, the auctioneer decides for the English. A third case is where many sellers and many buyers are interested in buying and respectively selling similar items. In that case the auctioneer decides for a double auction, because it minimizes the number of messages sent and allows more than one item to be purchased with different prices to different parties.

4.2 Comparison of the Coordination Models

In addition to input variables defined in Section 4.1, we are going to use ι, for the number of items in the shopping list of an agent, λ, for the number of auctions created, and α_T, for the total number of buyer agents in the system (a large value represents stiff competition). According to these variables, we will calculate κ, the number of messages and μ, the number of agent migrations.

Single-Agent: In this coordination model, the buyer is represented with one agent, migrating from one auction to the next in order to buy the products on its shopping list. Because there is no other agent in the system negotiating on behalf of the buyer, there are no messages communicated ($\kappa = 0$).

Since every buyer is represented with a unique agent in the single-agent model, there is no coordination, and this one agent migrates for each item to all of the auctions related with the item until it gets the item. The number of migrations is calculated with:

$$\mu = (\lambda - 1) \cdot \left(1 - \frac{\alpha_b}{\alpha_T}\right) + 2$$

The two at the end of the equation stands for the migration of the agent to the first auction and from the last auction back to the buyer's node. $(\lambda - 1)$ comes from the fact that, in order to buy the item, it may need to migrate to all of the auctions, except the beginning auction. Assuming that all buyer agents have equal probability of obtaining an item, we multiply it with $\left(1 - \frac{\alpha_b}{\alpha_T}\right)$. Since, the increasing number of total agents increases the concurrency, the probability of the agent to get the item in the auctions decreases. In the best case there is no competition, the agent will get the item in the first auction, it joins. Therefore, without migrating to other auctions, it returns back to the buyer's node. In the worst case scenario, it is going to migrate to all the auctions in its itinere.

Tuple Spaces: In tuple spaces, each agent continuously asks the memory space for changes, which increases the number of messages extremely.

The number of agents in the tuple spaces model changes between the number of items and the number of auctions, $(\iota \leq \alpha_b \leq \lambda)$.

Let ϕ symbolize the frequency of tuple space querying, and δ the duration of an auction. The number of queries per agent to a tuple space during an auction is $\left(\frac{\delta}{\phi}\right)$. Every time there is a bid, a tuple in the tuple space changes, therefore, β messages are needed. In order to find the number of querying messages per auction for all the agents representing the buyer in all the auctions they join, we multiply $\left(\beta + \frac{\delta}{\phi}\right)$ with $\lambda \cdot \alpha_b$. In the best case scenario, there is only one buyer agent for each item, $\alpha_b = \iota$, and it gets the item in the first auction it joins, $\lambda = 1$. Which gives us the following formula for the total number of messages in this best case scenario:

$$\kappa_{best} = \lambda \cdot \alpha_b \left(\beta + \frac{\delta}{\phi}\right) = 1 \cdot \alpha_b \left(\beta + \frac{\delta}{\phi}\right) = \iota \left(\beta + \frac{\delta}{\phi}\right)$$

In the worst case $(\alpha_b \ll \alpha_T)$, the agents have to join all the auctions offering the item they want to buy. If $\alpha_b = \lambda$, then the formula for the number of messages will be follows:

$$\kappa_{worst} = \lambda \cdot \alpha_b \left(\beta + \frac{\delta}{\phi}\right) = \lambda \cdot \lambda \left(\beta + \frac{\delta}{\phi}\right) = \lambda^2 \left(\beta + \frac{\delta}{\phi}\right)$$

We found a formula that gives the number of messages for all competition scenarios as following. When $(\alpha_b \ll \alpha_T)$, it implies that $(\alpha_b/\alpha_T \approx 0)$, and when $(\alpha_b \approx \alpha_T)$, it implies that $(\alpha_b/\alpha_T \approx 1)$. Considering these, the following formula gives the number of messages for tuple spaces:

$$\kappa = \left(\frac{\alpha_b}{\alpha_T} \cdot \iota + \left(1 - \frac{\alpha_b}{\alpha_T}\right) \cdot \ \alpha_b \cdot \lambda\right) \left(\beta + \frac{\delta}{\phi}\right)$$

The number of migrations reaches its minimum value, when $(\alpha_b = \lambda)$, because in that case every agent goes to a different auction and since all auctions are visited, there is no need for migration from one auction to another. Otherwise, $(\alpha_b < \lambda)$, and the agents have to migrate to all the auctions, offering the item they want to buy $(\lambda - \alpha_b)$. Assuming that all buyer agents have equal probability of obtaining an item, we multiply the number of migrations with $\left(1 - \frac{\alpha_b}{\alpha_T}\right)$. Furthermore, each agent migrates to the first auction in its itinerary and back from the last auction in its itinerary, which denotes $2\alpha_b$ migrations. Considering these, the formula for the total number of migrations is:

$$\mu = (\lambda - \alpha_b) \left(1 - \frac{\alpha_b}{\alpha_T}\right) + 2\alpha_b$$

Publish and Subscribe: Publish and Subscribe: The Publish and Subscribe coordination model works very similar to tuple spaces. The only difference concerning the number of messages and the number

of migrations is that in publish and subscribe model, the agents do not query the tuple space regularly, but the tuple space sends a broadcast to the agents whenever there is a change.

Because of these similarities the formula for the number of migrations, which we have described in the tuple spaces, is the same for publish and subscribe and the formula for the number of messages is slightly modified. So the number of messages is calculated with:

$$\kappa = \left(\frac{\alpha_b}{\alpha_T} + \left(1 - \frac{\alpha_b}{\alpha_T}\right) \cdot \lambda\right) \cdot \beta(1 + \ \alpha_b)$$

Hierarchical Coordination: In this model, the agents associated with a buyer are organized in the form of a tree, with the root agent representing the buyer. The root has ι children, each of which is an agent for some product in the shopping list. At the second level each agent has children agents, which are responsible for one or many auctions for the item that the parent is responsible for. To be more precise, if a is a node with depth two, then it is the agent that represents the buyer in an auction relation with the item list, parent(a) is responsible for.

In this model, every leaf node sends a message to its parent and siblings after each bid of its corresponding auction. Other agents are not to be informed, since they take role in the auctions for different items.

Let $\lambda_{(b,i)}$ be the number of auctions buyer b joins related with item i. The worst case performance in terms of the number of messages transmitted arises, when $\alpha_b \ll \alpha_T$ and the buyer agent joins all the auctions. In this special case $\sum_{i=1}^{\iota} \lambda_{(b,i)} = \lambda_b$. Since, for each bid every agent sends a message to its parent and siblings, and there exists $\lambda_{(b,i)}$ siblings; number of messages transmitted is

$$\kappa = \sum_{i=1}^{\iota} \left(\lambda_{(b,i)}\right)^2 \cdot \beta$$

From these, we can express worst case performance as the solution of the following quadratic convex program:

$$sup \ \sum_{i=1}^{\iota} \left(\lambda_{(b,i)}\right)^2 \cdot \beta$$

$$st \ \sum_{i=1}^{\iota} \left(\lambda_{(b,i)}\right) = \lambda_b \ and \ \lambda_{(b,i)} \geq 0 \ \ \forall i \in \{1, 2, ..., \iota\}$$

Supremum of the above problem is $(\lambda_b)^2 \cdot \beta$, and it is achieved, when $\lambda_{(b,i)} = \lambda_b$ for some $i \in (1, 2, ..., \iota)$ and $\lambda_{(b,i')} = 0, \ \forall i' \in (1, 2, ..., \iota) \setminus \{i\}$. Then:

$$\kappa_{worst} = (\lambda_b)^2 \cdot \beta$$

In the best case, the agents get the items in the auction they first join, therefore the total number of auctions joined is equal to ι (one auction per item in the shopping list). The number of total messages is obtained, when $\lambda_{(b,i)} = 1$, $\forall i \in \{1, 2, ...\iota\}$.

$$\kappa_{best} = \Sigma_{i=1}^{\iota} (\lambda_{(b,i)})^2 \cdot \beta = \Sigma_{i=1}^{\iota} \beta = \iota \cdot \beta$$

For the average case; let α_i be the number of buyers desiring to buy item i; let λ_i be the number of auctions related with item i. Expected number of messages between the agents of the buyer is given by

$$E\left[\kappa\right] \ = \ E\left[\sum_{i=1}^{\iota} (\lambda_{(b,i)})^2 \cdot \beta\right] \ = \ \beta \cdot E\left[\sum_{i=1}^{\iota} (\lambda_{(b,i)})^2\right] = \beta \cdot \sum_{i=1}^{\iota} E\left[(\lambda_{(b,i)})^2\right]$$

Assuming $\lambda_i \leq \alpha_i$, every agent participating in the auction has an equal probability of winning, and the winner agent does not join the remaining auctions related with i^{th} item,

$$E\left[(\lambda_{(b,i)})^2\right] = \sum_{j=1}^{\lambda_i} \frac{j^2}{\alpha_i} = \frac{1}{\alpha_i} \cdot \sum_{j=1}^{\lambda_i} j^2 = \frac{\lambda_i(\lambda_i + 1)(2\lambda_i + 1)}{6\alpha_i}$$

Therefore;

$$E[\kappa] = \beta \cdot \sum_{i=1}^{\iota} \frac{\lambda_i(\lambda_i + 1)(2\lambda_i + 1)}{6\alpha_i}$$

In the hierarchical coordination model the number of migrations is equal to the number of edges in the tree, which is one less than number of nodes in the tree. Therefore;

$$\mu = \left(1 + \iota + \sum_{i=1}^{\iota} \lambda_{(b,i)}\right) - 1 = \iota + \sum_{i=1}^{\iota} \lambda_{(b,i)}$$

Under the assumptions in the analysis of the expectation of κ

$$E[\mu] = E\left[\iota + \sum_{i=1}^{\iota} \lambda_{(b,i)}\right] = \iota + \sum_{i=1}^{\iota} E[\lambda_{(b,i)}] = \iota + \sum_{i=1}^{\iota} \sum_{j=1}^{\lambda_i} \frac{j}{\alpha_i} = \iota + \sum_{i=1}^{\iota} \frac{\lambda_i(\lambda_i + 1)}{2\alpha_i}$$

Comparing these four coordination models, we reach the conclusion that the hierarchical coordination model gives the best results for large-scale systems. Because of its scalability property, it minimizes the number of remote messaging by creating groups of agents. Therefore it allows minimal number of messages and migrations for large number of agents.

4.3 Software Simulation and Experimental Results

The software simulation presented shows something that an analysis cannot: time measurements. Thus, the goal of the prototype is to validate the analytical model and to provide insights into the significance of remote communication versus migration in terms of time. Furthermore, relationships between number of agents and different variables can be measured. The software prototype of our electronic marketplace and the experimental results are described in the following paragraphs.

Software Prototype

Our electronic marketplace prototype implements the open-exit English auction (see Section 2.1) with a single-agent approach coordination model (see Section 2.2). The SALSA[22] programming language was used to implement it. The language and system take advantage of the concurrency and migration characteristics of the actor model. Given enough bidding interaction, migration reduces network usage by avoiding remote messaging.

The system is a simplified version of the proposed model. Stores can join the system dynamically. Stores spawn auctions regularly. Buyer agents are created dynamically in the system. An itinerary of stores to visit is created as the buyer agents join the system. Each agent has a time to return. This time is divided evenly among stores in the itinerary. Therefore, buyer agents migrate from one store to the next in their itinerary. Once an agent migrates to a store, it joins relevant auctions. The buyer agents try to fulfill the shopping list of their buyer.

Experimental results

In this section, we will try to obtain the amount of time, the number of messages needed, the number of bids and the price to complete a transaction. We will also evaluate how these variables change when the number of buyer agents increase in different contexts. Three scenarios are as follows:

- **A demand driven world:** The supply of items is not enough to match the demand.
- **A supply driven world:** The supply of items is more than enough to match the demand.
- **A balanced supply and demand world:** The supply and demand of items are balanced.

All of these scenarios are evaluated with fixed characteristics such as budget and time in the system for buyers. A characteristic that is not fixed is the number of stores, which grows with the number of agents to make the bidding process more fair.

The results of testing the prototype are summarized in Figure 3. The upper-left graph plots the behavior of number of messages per transaction as the number of agents increases. The upper-right graph plots the

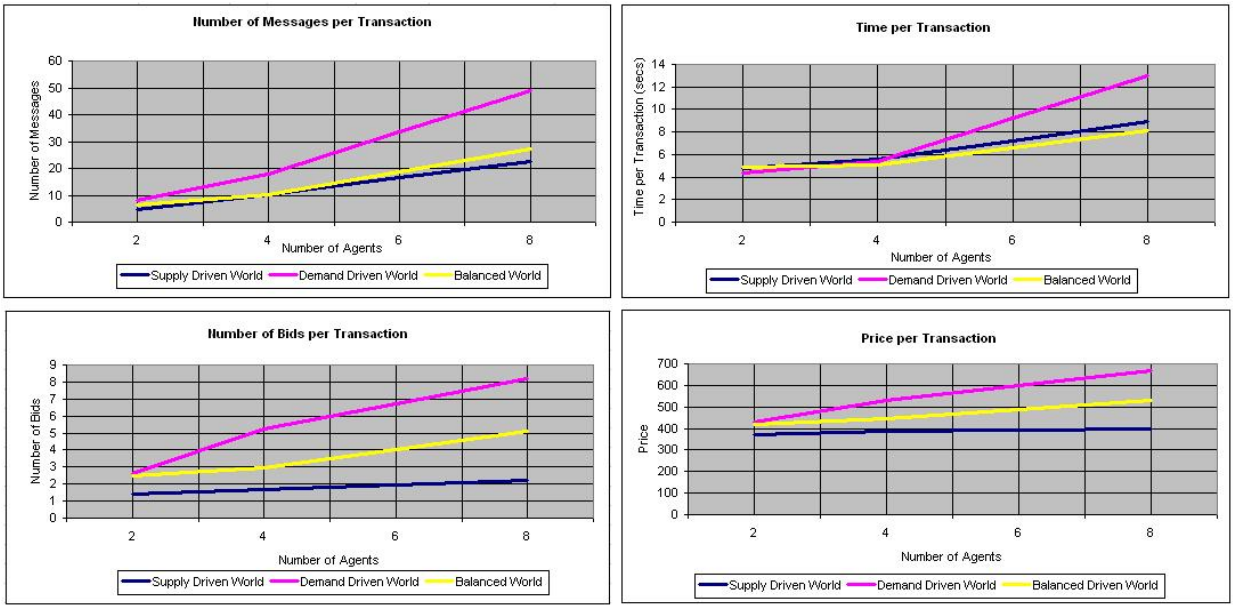


Figure 3: Messages, bids, time and price needed to complete a transaction as the number of agents grow.

behavior of time per transaction as the number of agents increases. The lower-left graph plots the behavior of the number of bids per transaction as the number of agents increases. The lower-right graph plots the behavior of the price as the number of agents increases.

Number of messages and bids per transaction: In a supply driven world, the number of messages slightly increases as the number of agents increases. This is because the probability of competition grows as the system becomes more complex. As probability of competition grows, so does the number of bids and therefore the number of messages. The most critical case is the demand driven world: lots of competition among agents leads to exponential increase in messages. This was predicted in the analytical model of Section 4.1. In a balanced supply-demand world, the number of bids to get an item tends to increase in a linear fashion, and so does the number of messages.

Time spent per transaction: In a supply driven world as more agents join the system, more stores offer the item desired. Therefore time per transaction increases since there are more stores to migrate to in order to fullfill the itinerary of the agent. In a demand driven world the time per transaction increases due to competition. For buyer agent to complete a transaction, the number of bids increases, and so does the time. Likewise in a balanced supply-demand world, the amount of time needed to complete a transaction increases.

Price per transaction: Since the number of stores for this simulation is a function of the number of agents, the price per transaction tends to increase slightly. In a demand driven world, in the case where the number of stores were fixed, the prices per transaction would be a lot higher than the prices obtained.

Migration vs. Remote Messaging

Two approaches to implement auction systems are migration and remote messaging. In the former, the parties of the auction are in the same location, buyer agents migrate to the seller agents location where the auction takes place. In the latter, the bidding process is held through remote messaging. The prototype proposed fits in the first approach while eBay-like systems fit in the second. To be able to come up with time measurements we introduce Table 2.

Table 2 (taken from [22]) shows ranges of values measured with minimal actors and empty messages, as well as larger actors. This values are used to show different aspects of remote comunication and migration. These numbers were taken with a testbed containing three LANs and one WAN. The WAN configuration was

Local actor creation	386μs
Local message sending	148μs
LAN message sending	30-60ms
WAN message sending	2-3 secs
LAN minimal actor migration	150-160ms
LAN 100Kb actor migration	240-250ms
WAN minimal actor migration	3-7secs
WAN 100Kb actor migration	25-30secs

Table 2: Local, LAN and WAN Time Ranges [22]

tested with three machines: one in Europe, one in USA and one in Japan. Local times are in microseconds, LAN times are in milliseconds, and WAN times are in seconds.

In eBay-like systems, the cost of comunication is very expensive. All the messaging is remote. Consider a scenario in which a buyer agent wants to buy two items. The agent is in a supply driven world context. An english auction is being held and the buyer agent is the only one participating. From the simulation results, an average of four and a half messages are sent to complete one transaction. The bidding process will be very simple, since there are no messages generated after outbids. To complete two transactions, there is a need of nine messages.

In an eBay-like system, nine remote messages are all that is needed. In a LAN environment, the time to complete nine messages is 270-540ms. In our prototype, nine local messages are needed and two migrations. In a local environment, the time to complete nine messages is 1332μs, or 1,3ms. The two LAN migrations are completed in 300-320ms. The time to buy two items would be 301,3 to 321,3ms. Therefore in LAN environments the two systems would perform similar as far as time.

The eBay-like systems are widely used in WAN environments. In a WAN environment nine messages will be completed after 18-27 seconds. In our prototype, the messaging would cost the same, but the migration would change. Two migrations in a WAN environment would cost 6-14 seconds. In the worst case of our prototype performance, the time would be less than in the best case of the eBay-like system.

Considering that eBay-like systems are more complex than our example scenario, the migration tool would improve the fairness of the bidding process. The bandwith and latency would no longer be a problem. It is also true that the bidding decision is taken in microseconds. Therefore the system must become autonomous. Human interaction in terms of user preferences will only be needed once at the beginning and once at the end of the process.

5 DISCUSSION AND FUTURE WORK

In this paper, we have considered a futuristic electronic marketplace consisting of software agents representing human buyers, sellers, and auctioneers.

We studied and analyzed different types of auctions in terms of their potential to satisfy buyers, sellers, and/or auctioneers, and in terms of their electronic complexity, that is, number of software agents, cost and patterns of communication, and likelihood of inducing transactions. English auctions favor sellers in terms of profit, particularly on demand-driven scenarios. Reverse auctions favor buyers in terms of budget, particularly on supply-driven scenarios. Sealed auctions have minimum requirements on messaging which favors environments with expensive network communication–latency and bandwidth. Double auctions provide the most fairness to buyers and sellers in terms of exchange prices, and maximizes transactions–a key auctioneer metric, yet incurs in the most communication overhead and can therefore significantly benefit from agent co-locality.

We also studied and analyzed different multi-agent coordination models in terms of computational complexity, and electronic commerce metrics, such as user satisfaction. While a buyer could have a single agent to accomplish its shopping task, thereby requiring no coordination, this strategy diminishes the probability

of success getting all the items desired, or getting the best prices. This is particularly true upon fierce competition from other buyers. Therefore, we analyzed the number of agents and messages required by three specific coordination models: tuple spaces, publish and subscribe model, and hierarchical coordination, with the goal of maximizing the probability of user satisfaction.

Finally, we prototyped an actor system to simulate an electronic marketplace, and we provided interesting empirical results that validate the approach of dynamically choosing auction types and buyer agent coordination models.

Further work involves considering empirically obtained actor creation, actor migration, and remote communication times to feed back into the analytical models to help determine the best strategy not just in terms of message passing and number of agents, but also in terms of communication and coordination times on a given network infrastructure. The most informed and best coordinated buyer agent teams should outperform other buyer agents in terms of meaningful electronic commerce metrics such as number of successful transactions and purchase prices as compared to average purchase prices by other (human and otherwise) buyers.

Ultimately, the goal is to create an adaptive electronic marketplace that can successfully complement today's commercial activities and bring in a new perspective to the fair and efficient exchange of goods.

References

[1] G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, 1986.

[2] G. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. A foundation for actor computation. *Journal of Functional Programming*, 7:1–72, 1997.

[3] Marcos Kawazoe Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar Deepak Chandra. Matching events in a content-based subscription system. In *PODC*, pages 53–61, 1999.

[4] T. Berners-Lee, R. Cailliau, A. Luotenen, H. F. Nielsen, and A. Secret. The World-Wide Web. *Communications of the ACM.*, 37(8), Aug 1994.

[5] C. Callsen and G. Agha. Open Heterogeneous Computing in ActorSpace. *Journal of Parallel and Distributed Computing*, pages 289–300, 1994.

[6] N. Carriero and D. Gelernter. *How to Write Parallel Programs*. MIT Press, 1990.

[7] R.A. Feldman and R. Mehra. Auctions theory and applications. *IMF Staff Papers*, pages 485–511, 1993.

[8] D. Friedman. The double auction market: Institutions, theories, and evidence. In D. Friedman and J. Rust, editors, *Proceedings of the Workshop on Double Auction Markets*, Santa Fe Institute Studies in the Sciences of Complexity, pages 3–25, Cambridge, 1993. Perseus Publishing.

[9] J. Hendler, T. Berners-Lee, and E. Miller. Integrating applications on the semantic web. *Journal of the Institute of Electrical Engineers of Japan*, 122(10):676–680, October 2002.

[10] S.D. Jap. An exploratory study of the introduction of online reverse auctions. *Journal of Marketing*, 67(4):118–122, October 2003.

[11] P. Milgrom. Auctions and bidding: A primer. *Journal of Economic Perspectives*, 3:3–22, 1989.

[12] Associated General Contractors of America. Associated General Contractors of America White Paper on Reverse Auctions for Procurement of Construction. January 2001. www.agc.org/content/public/pdf/Member_Resources/ReverseAuctionWhitePaper.pdf.

[13] David C. Parkes and Lyle H. Ungar. Iterative combinatorial auctions: Theory and practice. In *AAAI/IAAI*, pages 74–81, 2000.

[14] K. Reynolds. Auction types: Dutch, 1996. http://www.agorics.com/Library/Auctions/auction3.html.

[15] K. Reynolds. Auction types: English, 1996. http://www.agorics.com/Library/Auctions/auction2.html.

[16] K. Reynolds. Auctions: First-price, sealed bid; discriminatory, 1996. http://www.agorics.com/Library/Auctions/auction4.html.

[17] K. Reynolds. The double auction, 1996. http://www.agorics.com/Library/Auctions/auction6.html.

[18] H. A. Simon. *The Sciences of the Artificial*, chapter The Architecture of Complexity: Hierarchic Systems. MIT Press, 3rd edition, 1996.

[19] R. Tenorio and R.F. Easley. Bidding strategies in internet yankee auctions: Theory and evidence. *IEEE Parallel and Distributed Technology*, March 1991. http://ideas.repec.org/p/sce/scecf9/1021.html.

[20] C. Varela. *Worldwide Computing with Universal Actors: Linguistic Abstractions for Naming, Migration, and Coordination.* PhD thesis, U. of Illinois at Urbana-Champaign, 2001. http://osl.cs.uiuc.edu/Theses/varela-phd.pdf.

[21] C. Varela and G. Agha. A Hierarchical Model for Coordination of Concurrent Activities. In P. Ciancarini and A. Wolf, editors, *Third International Conference on Coordination Languages and Models (COORDINATION '99)*, LNCS 1594, pages 166–182, Berlin, April 1999. Springer-Verlag. http://osl.cs.uiuc.edu/Papers/Coordination99.ps.

[22] Carlos Varela and Gul Agha. Programming dynamically reconfigurable open systems with SALSA. *ACM SIGPLAN Notices. OOPSLA'2001 Intriguing Technology Track Proceedings*, 36(12):20–34, December 2001. http://www.cs.rpi.edu/~cvarela/oopsla2001.pdf.

[23] M. Vetter and S. Pitsch. An agent-based market supporting multiple auction protocols. In *Workshop on Agents for Electronic Commerce and Managing the Internet-Enabled Supply Chain*. Third International Conference on Autonomous Agents, 1999.

[24] M. Vetter and S. Pitsch. Towards a flexible trading process over the internet. In Frank Dignum and Carles Sierra, editors, *AgentLink*, volume 1991 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2001.

[25] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, March 1961.

Equipo Elitista de Algoritmos Evolutivos Multiobjetivo

José Manuel Fernandez Giangreco
Ciencias y Tecnología - Universidad Católica Nuestra Señora de la Asunción
Centro Nacional de Computación - Universidad Nacional de Asunción
Asunción, Paraguay
jmfernandez@cnc.una.py

Benjamín Barán
Ciencias y Tecnología - Universidad Católica Nuestra Señora de la Asunción
Centro Nacional de Computación - Universidad Nacional de Asunción
Asunción, Paraguay
bbaran@cnc.una.py

Resumen

Con el uso cada vez más extendido de Algoritmos Evolutivos para Optimización Multiobjetivo en problemas del mundo real, se hace necesario mejorar su desempeño sacando el máximo provecho de las bondades de cada algoritmo. Para ello, una reconocida alternativa es la incorporación de conceptos de paralelismo. En consecuencia, los algoritmos evolutivos multiobjetivo paralelos se constituyen en un área de creciente interés para aplicaciones prácticas, sobre todo en problemas industriales, financieros y de ingeniería aplicada.

Dado que estos algoritmos difieren en su desempeño según el tipo del problema, el presente trabajo propone un Equipo Elitista de Algoritmos Evolutivos Multiobjetivo que combina diversos algoritmos en el contexto de la Computación Paralela asíncrona. Resultados experimentales demuestran las ventajas (y desventajas) de esta nueva propuesta, al compararla con implementaciones paralelas de diversos algoritmos evolutivos multiobjetivos ya publicados.

Palabras Claves: Algoritmos Evolutivos, Optimización Multiobjetivo, Paralelismo, Equipo de Algoritmos.

Abstract

With the extended use of Evolutionary Algorithms for Multiobjective Optimization in real world problems, it becomes necessary to improve their performance taking advantage of each algorithm virtue. For this purpose, a well-known alternative is the incorporation of parallelism. Therefore, parallel multiobjective evolutionary algorithms are becoming an area of growing interest for practical applications, mainly in industrial, financial and applied engineering problems.

Since these algorithms differ in performance for different kind of problems, an Elitist Team Algorithm combining different Multiobjective Evolutionary Algorithms is proposed in a Parallel Computation context. Experimental results validate this new proposal showing several advantages (and a few disadvantages) when it is compared to parallel implementations of diverse well-known multiobjective evolutionary algorithms.

Keywords: Evolutionary Algorithms, Multiobjective Optimization, Parallelism, Team Algorithm.

1. Introducción

En la búsqueda de soluciones a problemas del mundo real puede ser necesario satisfacer de manera simultánea múltiples objetivos, los cuales suelen ser contradictorios [22]. De existir la posibilidad de combinar los diferentes objetivos y conociendo la mejor manera de hacerlo, se puede simplemente considerar la existencia de un único objetivo a optimizar, resultante de la combinación de todos los objetivos considerados. Sin embargo, lo usual es desconocer de que manera se deben combinar los diferentes objetivos, o esta combinación resulta inadecuada, cuando no imposible. Entonces, se trata de un Problema de Optimización Multiobjetivo (*Multiobjective Optimization Problem* - MOP) [8, 20, 22, 26].

En los problemas de optimización multiobjetivo con objetivos contradictorios no existe una solución única que pueda ser considerada la mejor, sino un conjunto de alternativas representando las mejores relaciones de compromiso entre todos los objetivos, en el sentido que cada solución es mejor que las otras en algún objetivo, pero ninguna es mejor que otra en todos los objetivos simultáneamente [22]. Dicho conjunto es llamado conjunto de soluciones Pareto óptimo y sus correspondientes vectores en el espacio objetivo constituyen el denominado Frente Pareto [20].

Los Algoritmos Evolutivos (*Evolutionary Algorithms* - EAs) [1] han demostrado ser especialmente adecuados para la optimización multiobjetivo [10, 11, 16, 19, 27]. En la actualidad, existe un gran número de Algoritmos Evolutivos para Optimización Multiobjetivo (*MultiObjective Evolutionary Algorithms* - MOEAs) [9]. Con el uso cada vez más extendido de estos algoritmos en problemas reales de optimización, se hace necesario mejorar el desempeño de los mismos [2, 18]. Por lo tanto, para asegurar la aplicabilidad de la técnica de optimización evolutiva multiobjetivo a problemas de complejidad creciente, es necesario mejorar tanto la efectividad como la eficiencia de los métodos evolutivos. Para ello, una alternativa es la incorporación de conceptos de paralelismo al diseño de estos algoritmos [18, 25], así tenemos a los algoritmos evolutivos multiobjetivo paralelos (*parallel Multiobjective Evolutionary Algorithms* - pMOEAs).

Por otra parte se puede considerar a los algoritmos en equipo (*Team Algorithm* - TA), que han demostrado ser una excelente técnica computacional para combinar una variedad de algoritmos corriendo en diferentes procesadores de una red típicamente asíncrona, como las actuales redes de área local conformadas por computadores personales heterogéneos [2, 3, 4, 5, 6, 7].

Ya que se dispone de diversos algoritmos pMOEAs, se sugiere aquí la conformación de un *Team Algorithm of Multiobjective Evolutionary Algorithms* (TA-MOEA) como una alternativa válida para mejorar tanto la efectividad como la eficiencia, al poder explorar diferentes porciones del espacio de búsqueda, posiblemente disjuntas, con características variadas proporcionadas por algoritmos diferentes.

Siendo el área de los algoritmos evolutivos multiobjetivo un área de creciente importancia y reciente interés, este trabajo propone determinar las ventajas y desventajas existentes en el desarrollo y aplicación de un TA-MOEA que combine las mejores características de cada MOEA, conformando un TA paralelo de algoritmos evolutivos multiobjetivo. A fin de determinar la efectividad de la técnica propuesta, inspirado en [20, 26, 30], el presente trabajo realiza una comparación de distintos pMOEAs y el TA-MOEA utilizando diferentes métricas experimentales usualmente utilizadas para medir su desempeño [17, 26].

Entonces, el presente trabajo propone un TA-MOEA que utiliza un criterio de selección elitista de algoritmos, conforme se explicará en detalle en la Sección 4, por el cual se va reemplazando al peor MOEA de un conjunto de algoritmos trabajando en equipo, por otro MOEA de mejor desempeño en el referido equipo de algoritmos. Para determinar el mejor y el peor MOEA de un equipo de algoritmos trabajando en paralelo sobre un mismo problema, se realiza un *ranking* de los pMOEAs, basado en el número de soluciones aportadas al equipo (TA) por cada algoritmo.

El presente trabajo está organizado de la siguiente manera: en la Sección 2 se presenta la definición general de un problema de optimización multiobjetivo. La formulación de los problemas ZDT con sus funciones objetivo es dada en la Sección 3. Los algoritmos en equipo son explicados en la Sección 4. Los resultados experimentales son mostrados en la Sección 5. Por último, las conclusiones y los trabajos futuros son presentados en la Sección 6.

2. Optimización Multiobjetivo

Un Problema de Optimización Multiobjetivo (MOP) general incluye un conjunto de n parámetros (variables de decisión), un conjunto de k funciones objetivo, y un conjunto de m restricciones. Las funciones objetivo y las restricciones son funciones de las variables de decisión. Luego, un MOP puede expresarse como:

$$\begin{array}{ll} \text{Optimizar} & \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})) \\ \text{sujeto a} & \boldsymbol{e}(\boldsymbol{x}) = (e_1(\boldsymbol{x}), e_2(\boldsymbol{x}), \ldots, e_m(\boldsymbol{x})) \geq \boldsymbol{0} \\ \text{donde} & \boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \boldsymbol{X} \\ & \boldsymbol{y} = (y_1, y_2, \ldots, y_k) \in \boldsymbol{Y} \end{array} \qquad \textbf{(2.1)}$$

siendo $\boldsymbol{x}$ el vector de decisión e $\boldsymbol{y}$ el vector objetivo. El espacio de decisión se denota por $\boldsymbol{X}$, y al espacio objetivo por $\boldsymbol{Y}$. Optimizar, dependiendo del problema, puede significar igualmente, minimizar o maximizar. El conjunto de restricciones $\boldsymbol{e(x)} \geq \boldsymbol{0}$ determina el conjunto de soluciones factibles $\boldsymbol{X}_f$ y su correspondiente conjunto de vectores objetivo factibles $\boldsymbol{Y}_f$.

El problema de optimización multiobjetivo consiste en hallar la $\boldsymbol{x}$ que tenga el "mejor valor" de $\boldsymbol{f(x)}$. En general, no existe un único "mejor valor", sino un conjunto de soluciones óptimas. Entonces, un nuevo concepto de optimalidad debe ser establecido para MOPs. Dados dos vectores de decisión $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{X}$, se tiene:

$$\begin{array}{ll} \boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{v}) & \text{si y solo si } \forall i \in \{1, 2, \ldots, \mathrm{k}\}: f_i(\boldsymbol{u}) = f_i(\boldsymbol{v}) \\ \boldsymbol{f}(\boldsymbol{u}) \preceq \boldsymbol{f}(\boldsymbol{v}) & \text{si y solo si } \forall i \in \{1, 2, \ldots, \mathrm{k}\}: f_i(\boldsymbol{u}) \leq f_i(\boldsymbol{v}) \\ \boldsymbol{f}(\boldsymbol{u}) \prec \boldsymbol{f}(\boldsymbol{v}) & \text{si y solo si } \boldsymbol{f}(\boldsymbol{u}) \preceq \boldsymbol{f}(\boldsymbol{v}) \wedge \boldsymbol{f}(\boldsymbol{u}) \neq \boldsymbol{f}(\boldsymbol{v}) \end{array} \qquad \textbf{(2.2)}$$

En un contexto de minimización, esta situación se expresa con los siguientes símbolos y términos:

$$\begin{array}{ll} \mathbf{u} \succ \mathbf{v} \ (\boldsymbol{u} \text{ domina a } \boldsymbol{v}) & \text{si y solo si } \boldsymbol{f}(\boldsymbol{u}) < \mathbf{f(v)} \\ \boldsymbol{v} \succ \boldsymbol{u} \ (\boldsymbol{v} \text{ domina a } \boldsymbol{u}) & \text{si y solo si } \boldsymbol{f}(\boldsymbol{v}) < \boldsymbol{f}(\boldsymbol{u}) \\ \boldsymbol{u} \sim \boldsymbol{v} \ (\boldsymbol{u} \text{ y } \boldsymbol{v} \text{ no son comparables}) & \text{si y solo si } \boldsymbol{f}(\boldsymbol{u}) \npreceq \boldsymbol{f}(\boldsymbol{v}) \wedge \boldsymbol{f}(\boldsymbol{v}) \npreceq \boldsymbol{f}(\boldsymbol{u}) \end{array} \qquad \textbf{(2.3)}$$

Dado un vector de decisión $\boldsymbol{x} \in \boldsymbol{X}_f$, se dice que $\boldsymbol{x}$ es no dominado respecto a un conjunto $\boldsymbol{V} \subseteq \boldsymbol{X}_f$ si y solo si $\boldsymbol{x} \succ \boldsymbol{v}$ o $\boldsymbol{x} \sim \boldsymbol{v}$, $\forall \boldsymbol{v} \in \boldsymbol{V}$. En caso que $\boldsymbol{x}$ sea no dominado respecto a todo el conjunto $\boldsymbol{X}_f$, y solo en ese caso, se dice que $\boldsymbol{x}$ es una solución Pareto óptima. Por lo tanto, el conjunto Pareto óptimo $\boldsymbol{X}_{true}$ puede ser definido formalmente como:

$$\boldsymbol{X}_{true} = \{ \boldsymbol{x} \in \boldsymbol{X}_f \mid \boldsymbol{x} \text{ es no dominado con respecto a todo el conjunto } \boldsymbol{X}_f \} \qquad \textbf{(2.4)}$$

El correspondiente conjunto de vectores objetivo $\boldsymbol{Y}_{true} = \boldsymbol{f}(\boldsymbol{X}_{true})$ constituye el Frente Pareto óptimo [12].

Usualmente, $\boldsymbol{Y}_{true}$ no puede ser calculado en forma exacta en tiempos razonables por lo que para fines prácticos, se intentará calcular al menos una buena aproximación al Frente Pareto óptimo, conocido en la literatura como $\boldsymbol{Y}_{known}$ [26].

3. Problemas ZDT

Los Problemas ZDT constituyen un conjunto de seis problemas desarrollados por Zitzler, Deb, y Thiele [30] como un escenario completo de prueba (*test bed*) que contempla distintas posibilidades y dificultades al considerar problemas multiobjetivos [15]. Todos estos problemas del *test bed* son propuestos en un contexto de minimización y están estructurados de igual manera sobre tres funciones $\boldsymbol{f_1}$, $\boldsymbol{g}$ y $\boldsymbol{h}$:

$$\begin{array}{ll} \textit{Minimizar} & \boldsymbol{F}(\boldsymbol{x}) = (\boldsymbol{f_1}(x_1), \boldsymbol{f_2}(\boldsymbol{x})) \\ \textit{sujeto a} & \boldsymbol{f_2}(\boldsymbol{x}) = \boldsymbol{g}(x_2, \ldots, x_n) \cdot \boldsymbol{h}(\boldsymbol{f_1}(x_1), \boldsymbol{g}(x_2, \ldots, x_n)) \\ \textit{donde} & \boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \boldsymbol{X} \end{array} \qquad \textbf{(3.1)}$$

La función $\boldsymbol{f_1}$ es una función que depende únicamente de la primera variable de decisión, $\boldsymbol{g}$ es una función de las n-1 variables de decisión restantes y los parámetros de $\boldsymbol{h}$ son los valores de las funciones $\boldsymbol{f_1}$ y $\boldsymbol{g}$. Las funciones de prueba difieren en estas tres funciones así como en el número de variables n y en los valores que éstas pueden tomar.

El problema ZDT1 tiene 30 variables en el rango [0,1]. Su frente de Pareto es convexo. Tiene un frente Pareto continuo y una distribución uniforme de soluciones a lo largo del frente, donde $\boldsymbol{x}=(x_1,...,x_n)$, n=30 y $x_i \in [0,1]$.

$$\boldsymbol{f_1}(\boldsymbol{x_1}) = \boldsymbol{x_1}\,;\, \boldsymbol{g}(\boldsymbol{x_2},\ldots,\boldsymbol{x_n}) = 1+9\cdot\sum_{i=2}^{n}\frac{\boldsymbol{x_i}}{(n-1)}\,;\, \boldsymbol{h}(\boldsymbol{f_1},\boldsymbol{g}) = 1-\sqrt{\frac{\boldsymbol{f_1}}{\boldsymbol{g}}} \qquad \textbf{(3.2)}$$

El problema ZDT2 tiene 30 variables en el rango [0,1]. Su frente de Pareto es no convexo. La distribución de soluciones a lo largo del frente de Pareto es uniforme, donde $\boldsymbol{x}=(x_1,...,x_n)$, n=30 y $x_i \in [0,1]$.

$$\boldsymbol{f_1}(\boldsymbol{x_1}) = \boldsymbol{x_1}\,;\; \boldsymbol{g}(\boldsymbol{x_2},\ldots,\boldsymbol{x_n}) = 1+9\cdot\sum_{i=2}^{n}\frac{\boldsymbol{x_i}}{(n-1)}\,;\boldsymbol{h}(\boldsymbol{f_1},\boldsymbol{g}) = 1-\left(\frac{\boldsymbol{f_1}}{\boldsymbol{g}}\right)^2 \qquad \textbf{(3.3)}$$

El problema ZDT3 tiene 30 variables en el rango [0,1]. Su frente de Pareto es discontinuo. La distribución de soluciones a lo largo del frente es uniforme, donde $\boldsymbol{x}=(x_1,...,x_n)$, n=30 y $x_i \in [0,1]$.

$$\begin{aligned}&\boldsymbol{f_1}(\boldsymbol{x_1}) = \boldsymbol{x_1}\,;\, \boldsymbol{g}(\boldsymbol{x_2},\ldots,\boldsymbol{x_n}) = 1+9\cdot\sum_{i=2}^{n}\frac{\boldsymbol{x_i}}{(n-1)}\\ &\boldsymbol{h}(\boldsymbol{f_1},\boldsymbol{g}) = 1-\sqrt{\frac{\boldsymbol{f_1}}{\boldsymbol{g}}}-\left(\frac{\boldsymbol{f_1}}{\boldsymbol{g}}\right)\boldsymbol{sin}(10\pi\,\boldsymbol{f_1})\end{aligned} \qquad \textbf{(3.4)}$$

El problema ZDT4 tiene 10 variables en el rango [0,1]. Su frente de Pareto es convexo. La complejidad de este problema es que contiene 21^9 frentes Pareto-óptimos locales [30] y, por lo tanto, prueba a los algoritmos evolutivos con relación a su capacidad de lidiar con la multimodalidad. En este caso $\boldsymbol{x}=(x_1,...,x_n)$, n=10, $x_1 \in [0,1]$, $x_2,...,x_n \in [-5,5]$.

$$\begin{aligned}&\boldsymbol{f_1}(\boldsymbol{x_1}) = \boldsymbol{x_1}\,;\, \boldsymbol{g}(\boldsymbol{x_2},\ldots,\boldsymbol{x_n}) = 1+10(n-1)+\sum_{i=2}^{n}\left(\boldsymbol{x_i}^2-10\,\boldsymbol{cos}(4\pi\,\boldsymbol{x_i})\right)\\ &\boldsymbol{h}(\boldsymbol{f_1},\boldsymbol{g}) = 1-\sqrt{\frac{\boldsymbol{f_1}}{\boldsymbol{g}}}\end{aligned} \qquad \textbf{(3.5)}$$

El problema ZDT5 utiliza funciones booleanas definidas sobre *strings*. Constituye un caso de problema *deceptivo* ya que la forma de la función auxiliar determina que la mayor parte del espacio de búsqueda se concentra cerca de óptimos locales, mientras que el óptimo global se halla relativamente aislado. Aquí tenemos $\boldsymbol{x}=(x_1,...,x_n)$, n=11, $x_1 \in \{0,1\}^{30}$, $x_2,...,x_n \in \{0,1\}^5$.

$$\begin{aligned}&\boldsymbol{f_1}(\boldsymbol{x_1}) = 1+u(\boldsymbol{x_1})\,;\, \boldsymbol{g}(\boldsymbol{x_2},\ldots,\boldsymbol{x_n}) = \sum_{i=2}^{n} v(u(\boldsymbol{x_i}))\;;\boldsymbol{h}(\boldsymbol{f_1},\boldsymbol{g}) = \frac{1}{\boldsymbol{f_1}}\\ &u(\boldsymbol{x_i}) = \text{"Número de unos en } \boldsymbol{x_i}\text{"}\\ &v(u(\boldsymbol{x_i})) = \begin{cases}2+u(\boldsymbol{x_i}) & \text{si}\quad u(\boldsymbol{x_i})<5\\ 1 & \text{si}\quad u(\boldsymbol{x_i})=5\end{cases}\end{aligned} \qquad \textbf{(3.6)}$$

El problema ZDT6 tiene 10 variables en el rango [0,1]. Su frente de Pareto es no convexo. La complejidad de este problema está dada por la combinación de la forma no convexa del frente de Pareto y la distribución no uniforme de soluciones a lo largo de él, donde $\boldsymbol{x}=(x_1,...,x_n)$, n=10, $x_i \in [0,1]$.

$$\begin{aligned}&\boldsymbol{f_1}(\boldsymbol{x_1}) = 1-e^{(-4x_1)}\,\boldsymbol{sin}^6(6\pi\,\boldsymbol{x_1})\,;\, \boldsymbol{g}(\boldsymbol{x_2},\ldots,\boldsymbol{x_n}) = 1+9\cdot\left(\sum_{i=2}^{n}\frac{\boldsymbol{x_i}}{(n-1)}\right)^{\frac{1}{4}}\\ &\boldsymbol{h}(\boldsymbol{f_1},\boldsymbol{g}) = 1-\left(\frac{\boldsymbol{f_1}}{\boldsymbol{g}}\right)^2\end{aligned} \qquad \textbf{(3.7)}$$

4. Algoritmos en Equipo

El desempeño de un algoritmo se mide generalmente por la calidad de los resultados obtenidos y por la rapidez en la respuesta. En este aspecto, los *Team Algorithms* han demostrado ser muy eficientes superando muchas veces en desempeño a los métodos tradicionales [3, 4, 5, 6]. Debido a estas ventajas en combinar diferentes métodos, el presente trabajo propone la combinación de siete Algoritmos Evolutivos Multiobjetivos:

- *Multiobjective Genetic Algorithm* (MOGA) [19].
- *Nondominated Sorting Genetic Algorithm* (NSGA) [24].
- *Niched Pareto Genetic Algorithm* (NPGA) [21].
- *Nondominated Sorting Genetic Algorithm II* (NSGA2) [13].
- *Controlled Elitist NSGA II* (CNSGA2) [14].
- *Strength Pareto Evolutionary Algorithm* (SPEA) [29].
- *Strength Pareto Evolutionary Algorithm II* (SPEA2) [28].

Se utilizaron estos algoritmos por considerarlos de los más representativos entre los MOEAs.

Los *Team Algorithm* han surgido como una novedosa herramienta que no solo mejora los tiempos de respuesta del algoritmo sino que además logra un efecto sinergético, conforme fuera experimentalmente comprobado [3, 4, 5, 6].

El modelo propuesto en el presente trabajo se denomina *Team Algorithm of pMOEAs* y está formado por un proceso Coordinador y siete procesos paralelos esclavos en los cuales se pueden ejecutar uno de los siete MOEAs disponibles.

El diagrama de flujo del proceso Coordinador se observa en la Figura 1. El proceso Coordinador primeramente crea las estructuras para almacenar los resultados provenientes de los distintos procesos MOEA en su población, luego se agrega a un grupo de trabajo e inicia cada uno de los procesos MOEA con sus parámetros específicos. Esto es implementado utilizando primitivas de comunicación de grupo, proveídas por librerías de paso de mensajes [23], lo que facilita la comunicación entre los distintos procesos utilizados.

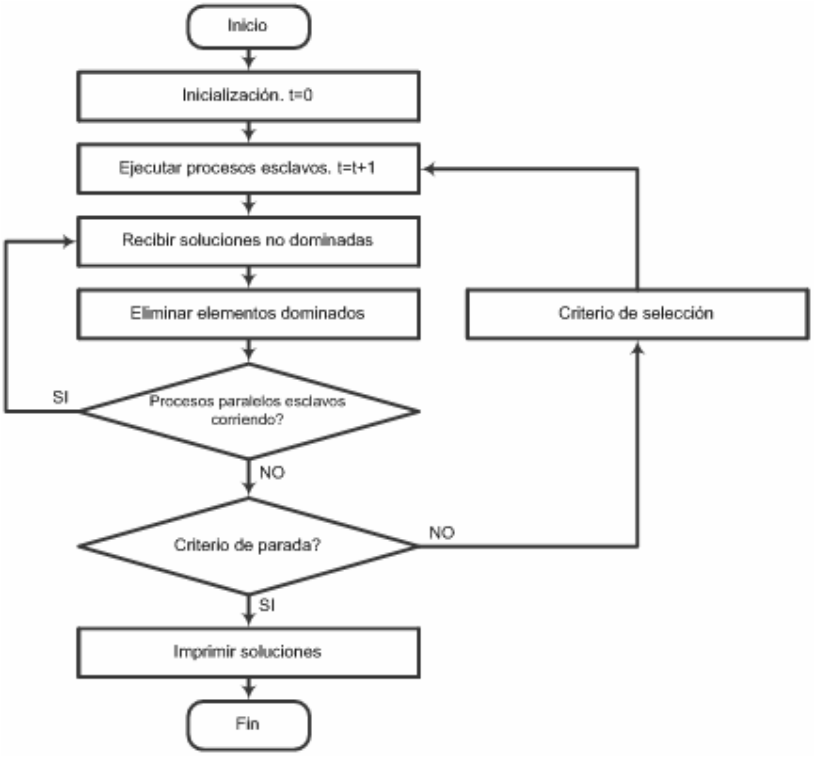


Figura 1. Diagrama de flujo del Coordinador

Durante la evolución, cada uno de los MOEAs que interviene en la búsqueda envía a todos los elementos que componen el equipo de trabajo un porcentaje de las mejores soluciones obtenidas. Cuando el Coordinador recibe estas soluciones las almacena en su población. A fin de mantener sólo las mejores soluciones, se eliminan las soluciones dominadas, también conocidas como soluciones cubiertas.

Cada MOEA se ejecuta independientemente en procesadores posiblemente diferentes, hasta cumplir un criterio local de parada. Cuando la cuenta de MOEAs en ejecución es igual a cero, el Coordinador verifica el criterio global de parada. De no ser alcanzado, se procede a seleccionar nuevos algoritmos esclavos para la siguiente iteración del TA-MOEA.

El criterio utilizado por el proceso Coordinador para la selección de los algoritmos esclavos es establecer un ranking de los algoritmos basado en el número de soluciones no dominadas aportadas a la población del Coordinador. El peor algoritmo es descartado y en su lugar se ubica una copia del mejor algoritmo de la última iteración.

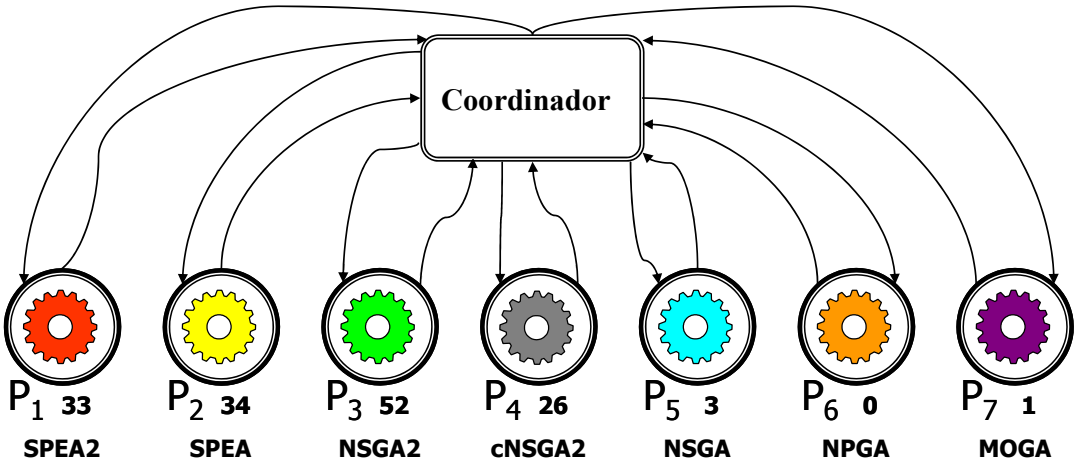


Figura 2. TA-MOEA con los MOEAs clasificados

En la Figura 2 se observan los siete procesos esclavos con los MOEAs correspondientes, clasificados conforme al número de soluciones encontradas. Así, tenemos que el mejor algoritmo es el NSGA2 con 52 soluciones aportadas a la población del Coordinador y el peor algoritmo es el NPGA con 0 soluciones aportadas.

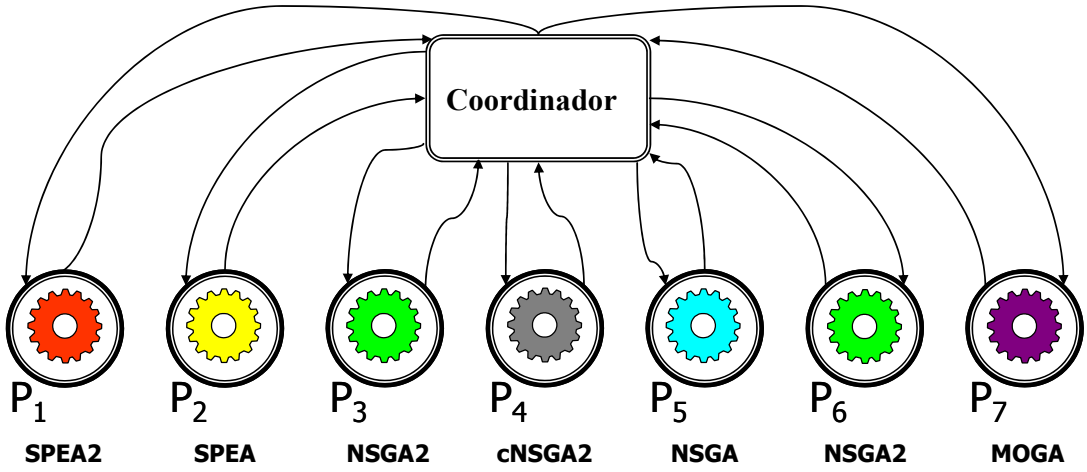


Figura 3. TA-MOEA luego del reemplazo

La Figura 3 presenta el TA-MOEA luego que una copia del NSGA2 reemplaza al NPGA.

Una vez seleccionados los algoritmos, se repite el ciclo hasta que el criterio de parada se cumpla; entonces, el Coordinador imprime todas las soluciones Pareto contenidas en su población y finaliza la ejecución.

Cabe resaltar que a medida que transcurran las iteraciones, el conjunto de procesos tiende a ejecutar el mismo algoritmo, y dicho algoritmo generalmente resulta el mejor MOEA del conjunto de algoritmos utilizados por el TA-MOEA en el problema específico que se está resolviendo. Por lo tanto, otro aporte indirecto que surge del TA-MOEA es la capacidad de identificar cual algoritmo de los combinados es el mejor MOEA para un determinado problema.

5. Resultados Experimentales

Los siete algoritmos elegidos para este trabajo fueron implementados, para la resolución paralela de los seis problemas de prueba presentadas en la Sección 3. Los algoritmos seleccionados se implementaron de acuerdo a la literatura original de referencia. Así, quedaron constituidos 8 algoritmos a ser comparados experimentalmente, 7 de ellos conformados por los pMOEAs enumerados en la Sección 4 y el TA-MOEA propuesto en el presente trabajo.

Característica	Descripción
Tipo de computadora	COW
Tipo de CPU	AMD 800MHz
Memoria	256 MB
Sistema Operativo	Red Hat Linux v7.3
Red de comunicación	Ethernet 100 Mbps
Librería de comunicación	PVM 3.4.5

Tabla 1. Características del ambiente computacional paralelo utilizado

Se utilizaron *Cluster Of Workstations* (COW) en el ambiente computacional que presenta la Tabla 1.

Se realizaron corridas con 1, 2, 4 y 8 procesadores. Las tablas de la Sección 5.3 presentan los resultados promedios de las cuatro corridas.

Para comparar la calidad de los resultados obtenidos por las 8 implementaciones realizadas, se utilizarán en el presente trabajo dos reconocidas métricas ampliamente utilizadas para la comparación de algoritmos evolutivos multiobjetivo [26], definidas a continuación.

5.1. Métricas Utilizadas

La selección de las métricas es una tarea crítica, que debe realizarse cuidadosamente, para evitar obtener resultados poco útiles. Es conveniente recordar que ningún criterio único puede dar una idea acabada del desempeño general de los MOEAs, ya que algunos se enfocan en la efectividad y otros en la eficiencia.

En el presente trabajo se utilizaron dos métricas: la Generación de vectores no dominados, que proporciona la cantidad de soluciones encontradas por el algoritmo y la Generación real de vectores no dominados, que da la cantidad de soluciones encontradas por el algoritmo y que no son dominadas por las soluciones de los demás algoritmos.

Generación de vectores no dominados (GVND). Esta métrica cuenta el número de soluciones en el frente Pareto calculado $\boldsymbol{Y}_{known}$. Se puede definir mediante la siguiente ecuación:

$$GVND \stackrel{\Delta}{=} | Y_{known} |_c \qquad \textbf{(5.1)}$$

donde $\left| \ \right|_c$ denota cardinalidad.

Generación real de vectores no dominados (GRVND). La métrica denominada generación real de vectores no dominados cuenta la cantidad de elementos en el frente Pareto calculado que en efecto pertenecen al frente Pareto óptimo:

$$GRVND \stackrel{\Delta}{=} \left| \left\{ y \mid y \in Y_{known} \quad \wedge \quad y \in Y_{true} \right\} \right|_c \qquad \textbf{(5.2)}$$

Para el cálculo de esta métricas que requiere el conocimiento del frente Pareto óptimo real, se utilizó como aproximación un conjunto de soluciones no dominadas con respecto al conjunto unión de todos los resultados obtenidos considerando la totalidad de las ejecuciones realizadas por cada una de las 8 implementaciones arriba mencionadas.

5.3. Resultados experimentales sobre los problemas ZDT

En la Figura 4 se grafica el tiempo en segundos requerido por las implementaciones realizadas para la resolución de los seis problemas ZDT en relación con el numero de procesadores utilizados. Como puede apreciarse al aumentar el número de procesadores disminuye el tiempo de ejecución.

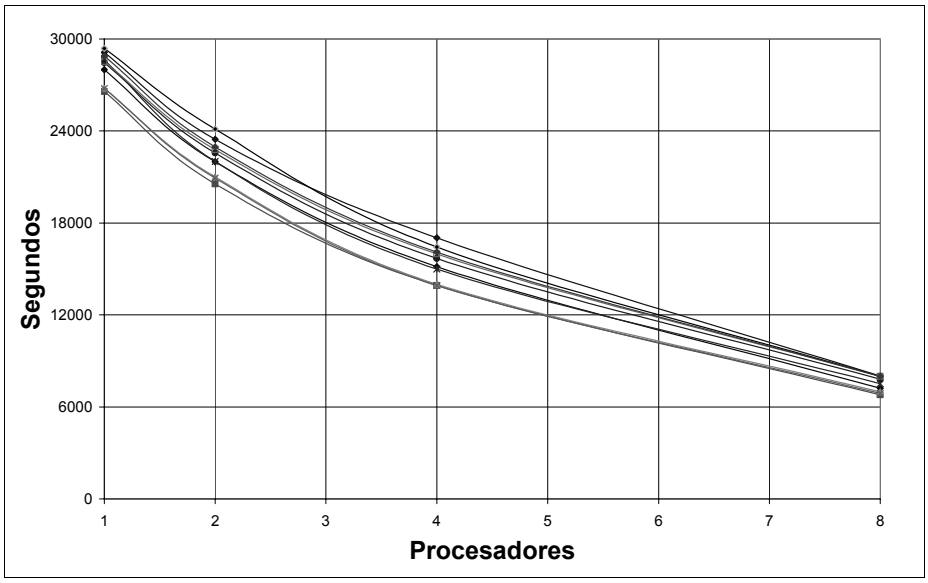


Figura 4. Tiempo total en segundos por número de procesadores

Basados en las métricas utilizadas, se procede a clasificar las 8 implementaciones realizadas, ordenándolas en un ranking (de mejor a peor) para cada problema ZDT.

	Conjunto	ZDT1
1	NSGA2	1150
2	TAMOEA	1079,75
3	SPEA	979
4	SPEA2	943
5	CNSGA2	795,5
6	NSGA	450
7	NPGA	429
8	MOGA	369,5

	Conjunto	ZDT2
1	SPEA	924
2	NSGA2	902
3	TAMOEA	901
4	SPEA2	860
5	CNSGA2	831
6	NPGA	567
7	NSGA	508
8	MOGA	498

	Conjunto	ZDT3
1	SPEA2	805
2	TAMOEA	801
3	NSGA2	761
4	CNSGA2	754,75
5	SPEA	753
6	NPGA	677,25
7	NSGA	550,75
8	MOGA	500,5

Tabla 2. Ranking considerando la métrica GVND sobre los problemas ZDT1, ZDT2 y ZDT3

En la Tabla 2 se observa las posiciones en los problemas ZDT1, ZDT2 y ZDT3 utilizando la métrica GVND. Para el problema ZDT1, el mejor fue el NSGA2 quedando en segundo lugar el TA-MOEA. En el problema ZDT2 quedó primero el SPEA mientras el TA-MOEA obtiene el tercer lugar. En el problema ZDT3, el SPEA2 obtuvo el primer lugar y el TA-MOEA quedó segundo.

	Conjunto	ZDT4
1	TAMOEA	892
2	SPEA2	875
3	SPEA	834
4	CNSGA2	791,25
5	NSGA2	782
6	MOGA	601
7	NPGA	590,75
8	NSGA	548

	Conjunto	ZDT5
1	SPEA2	854
2	NSGA2	853
3	SPEA	850,5
4	TAMOEA	830
5	CNSGA2	821
6	NPGA	580
7	NSGA	578
8	MOGA	532,25

	Conjunto	ZDT6
1	SPEA2	690
2	NSGA2	670
3	CNSGA2	662,75
4	TAMOEA	631,5
5	NPGA	582
6	NSGA	578
7	SPEA	547
8	MOGA	402

Tabla 3. Ranking considerando la métrica GVND sobre los problemas ZDT4, ZDT5 y ZDT6

En la Tabla 3 se observa las posiciones en los problemas ZDT4, ZDT5 y ZDT6 utilizando la métrica GVND. Como vemos para el problema ZDT4, el mejor fue el TA-MOEA. En el problema ZDT5, quedó primero el SPEA2 mientras el TA-MOEA obtiene el cuarto lugar. En el problema ZDT6 el SPEA2 obtiene el primer lugar y el TA-MOEA vuelve a quedar cuarto.

	Conjunto	ZDT1
1	NSGA2	940,25
2	TAMOEA	909,25
3	SPEA2	877,5
4	SPEA	801,75
5	CNSGA2	780,75
6	NSGA	3
7	NPGA	1
8	MOGA	0

	Conjunto	ZDT2
1	SPEA	866
2	NSGA2	820
3	TAMOEA	818
4	SPEA2	687
5	CNSGA2	659
6	NSGA	6
7	NPGA	2
8	MOGA	1

	Conjunto	ZDT3
1	SPEA2	788
2	TAMOEA	702
3	NSGA2	603
4	SPEA	603
5	CNSGA2	589,25
6	NSGA	58,75
7	NPGA	33,75
8	MOGA	7,25

Tabla 4. Ranking considerando la métrica GRVND sobre los problemas ZDT1, ZDT2 y ZDT3

En la Tabla 4 se observa las posiciones en los problemas ZDT1, ZDT2 y ZDT3 utilizando la métrica GRVND. Para el problema ZDT1, el mejor fue el NSGA2 seguido del TA-MOEA en segunda posición. En el problema ZDT2, quedó primero el SPEA mientras el TA-MOEA obtiene el tercer lugar. En el problema ZDT3 el mejor fue el SPEA2 mientras el TA-MOEA vuelve a quedar segundo.

	Conjunto	ZDT4
1	CNSGA2	701,5
2	SPEA	699
3	TAMOEA	690
4	SPEA2	657
5	NSGA2	602
6	NSGA	20
7	NPGA	12
8	MOGA	6

	Conjunto	ZDT5
1	SPEA2	632
2	CNSGA2	594
3	TAMOEA	587
4	SPEA	583,5
5	NSGA2	583
6	NSGA	23
7	NPGA	14
8	MOGA	7

	Conjunto	ZDT6
1	NSGA2	541
2	TAMOEA	501,25
3	SPEA	477
4	SPEA2	459
5	CNSGA2	430
6	NPGA	28
7	NSGA	23
8	MOGA	23

Tabla 5. Ranking considerando la métrica GRVND sobre los problemas ZDT4, ZDT5 y ZDT6

En la Tabla 5 se observa las posiciones en los problemas ZDT4, ZDT5 y ZDT6 utilizando la métrica GRVND. Ahora, en el problema ZDT4, el mejor es el CNSGA2, quedando en tercer puesto el TA-MOEA. En el problema ZDT5 quedó primero el SPEA2 mientras el TA-MOEA obtiene el tercer lugar. Por su parte, en el problema ZDT6 el primer lugar lo ocupa el NSGA2, quedando el TA-MOEA en segunda posición.

Generación de vectores no dominados (GVND)

	Conjunto	PROMEDIO
1	TAMOEA	855,875
2	NSGA2	853
3	SPEA2	837,833333
4	SPEA	814,583333
5	CNSGA2	776,041667
6	NPGA	571
7	NSGA	535,458333
8	MOGA	483,875

Generación real de vectores no dominados (GRVND)

	Conjunto	PROMEDIO
1	TAMOEA	701,25
2	SPEA2	683,416667
3	NSGA2	681,541667
4	SPEA	671,708333
5	CNSGA2	625,75
6	NSGA	22,2916667
7	NPGA	15,125
8	MOGA	7,375

Tabla 6. Posiciones promedio según la métrica GVND y la métrica GRVND

Haciendo un promedio global de los 6 problemas tratados, el TA-MOEA resulta claramente superior al considerar ambas métricas. En efecto, considerando la métrica GVND, obtuvo el primer lugar, seguido del NSGA2 y el SPEA2, como se aprecia a la izquierda de la Tabla 6. Por su parte, al considerar la métrica GRVND a la derecha de dicha figura, se observa que el primer lugar de nuevo es ocupado por el TA-MOEA, esta vez seguido del SPEA2 y el NSGA2.

Concluyendo esta sección, se puede afirmar que el TA-MOEA presenta experimentalmente soluciones más robustas ante un conjunto variado de problemas con características y dificultades diferentes, convirtiéndose en una opción válida para la resolución de problemas de optimización multiobjetivo, especialmente al considerar problemas nuevos con características no necesariamente conocidas.

6. Conclusiones y Trabajos Futuros

El presente trabajo propuso por primera vez una combinación de algoritmos evolutivos multiobjetivo en un contexto paralelo asíncrono utilizando un *Team Algorithm* implementado sobre una red de computadores.

A partir de los distintos resultados experimentales obtenidos, se han propuesto conclusiones parciales sobre la implementación del TA-MOEA con criterio de selección elitista, en la resolución de cada uno de los problemas considerados. Con esto, es posible arribar a las siguientes conclusiones finales de este trabajo:

- El diseño e implementación de un equipo de algoritmos evolutivos multiobjetivo paralelos es un problema complejo. Existen varias decisiones que tomar. El rango de éstas va desde el tipo de plataforma paralela en que se realizará la implementación, hasta la determinación de diversos parámetros de configuración y migración de datos.
- Teniendo en cuenta las métricas utilizadas para medir la calidad del conjunto Pareto calculado, se puede establecer que el TA-MOEA ha sido el que logró, en general, un mejor desempeño. Esta diferencia se verifica en toda la gama de problemas resueltos en este trabajo.
- Al utilizar TA-MOEA, se extiende el espacio de búsqueda. La recepción de elementos provenientes de distintos algoritmos introduce información genética en forma aleatoria que es útil para la obtención de mejores soluciones.

En resumen, a partir de las implementaciones consideradas en este trabajo, se recomienda el uso del Equipo Elitista de Algoritmos Evolutivos Multiobjetivo Paralelos (o TA-MOEA) por quedar mejor ubicado en el ranking global, teniendo en cuenta ambas métricas utilizadas, y además, por su capacidad de identificar razonablemente cual es el mejor MOEA para un determinado problema.

De forma a continuar con el trabajo iniciado, los siguientes tópicos son propuestos como trabajos futuros: implementación de TA-MOEAs que incorporen criterios de selección más refinados y dinámicos; así como el estudio e implementación de nuevos problemas de ingeniería que sirvan de base comparativa y experimental para el TA-MOEA.

Referencias

[1] T. Bäck, D. B. Fogel, y Z. Michalewicz, editores. "Handbook of Evolutionary Computation". Institute of Physics Publishing and Oxford University Press, 1997.

[2] B. Barán, J. Vallejos, R. Ramos, y U. Fernández. "Multi-objective reactive power compensation". En 2001 IEEE/PES Transmission and Distribution Conference and Exposition, volume 1, pg. 97–101. IEEE, 2001.

[3] B. Barán, E. Kaszkurewicz y D. M. Falcão. "Team Algorithms in Distributed Load Flow Computations". IEE Proceeding on Generation, Transmission and Distribution, Vol. 142, No. 6, pg. 583-588, noviembre 1995. Londres - Gran Bretaña.

[4] B. Barán, N. Cáceres y E. Chaparro. "Reducción del Tiempo de Búsqueda utilizando una Combinación de Algoritmos Genéticos y Métodos Numéricos". XV International Conference of the Chilean Computer Science Society. Arica - Chile, 1995.

[5] B. Barán, E. Kaszkurewicz y A. Bhaya. "Parallel Asynchronous Team Algorithms: Convergence and Performance Análisis". IEEE Transactions on Parallel & Distributed Systems, Vol. 7, No. 7, pg. 677-688, julio 1996. Estados Unidos.

[6] B. Barán, E. Chaparro y N. Cáceres. "A-Teams en la Optimización del Caudal Turbinado de una Represa Hidroeléctrica". IBERAMIA-98, Lisboa-Portugal. 1998

[7] B. Barán y F. Laufer. "Topological Optimization of Reliable Netwoks using A-Teams". Systemics, Cybernetics and Informatics SCI'99, Orlando - Florida, Estados Unidos. 1999.

[8] C. A. Coello Coello y G. Toscano Pulido. "A Micro-Genetic Algorithm for Multiobjective Optimization". En E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, y D. Corne, editores, First International Conference on Evolutionary Multi-Criterion Optimization, pg. 126–140. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[9] C. A. Coello Coello y C. E. Mariano Romero. "Evolutionary Algorithms and Multiple Objective Optimization". En M. Ehrgott y X. Gandibleux, editores, Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys, pg. 277–331. Kluwer Academic Publishers, Boston, 2002.

[10] C. A. Coello Coello. "An Updated Survey of GA-Based Multiobjective Optimization Techniques". Technical Report Lania-RD-98-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, México, December 1998.

[11] C. A. Coello Coello. "Constraint handling through a multiobjective optimization technique". En A. S. Wu, editor, Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program, pg. 117–118, Orlando, Florida, Julio 1999.

[12] J. Crichigno y B. Barán, "Multiobjective Multicast Routing Algorithm". IEEE ICT'2004, Ceará, Brasil, 2004.

[13] K. Deb, S. Agrawal, A. Pratab, y T. Meyarivan. "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II". KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[14] K. Deb y T. Goel. "Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergente". En E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, y D. Corne, editores, First International Conference on Evolutionary Multi-Criterion Optimization, pg. 67–81. Springer- Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[15] K. Deb. "Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems", Technical Report CI-49/98, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, 1998.

[16] K. Deb. "Evolutionary algorithms for multi-criterion optimization in engineering design". En K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, y J. Periaux, editores, Evolutionary Algorithms in Engineering and Computer Science, pg. 135–161, Chichester, UK, 1999. JohnWiley & Sons, Ltd.

[17] N. M. Duarte, A. E. Ruano, C. M. Fonseca, y P. J. Fleming. "Accelerating Multi-Objective Control System Design Using a Neuro-Genetic Approach". En 2000 Congress on Evolutionary Computation, volume 1, pg. 392–397, Piscataway, New Jersey, Julio 2000. IEEE Service Center.

[18] S. Duarte y B. Barán. "Multiobjective Network Design Optimisation Using Parallel Evolutionary Algorithms". En XXVII Conferencia Latinoamericana de Informática CLEI-2001, Mérida, Venezuela, 2001.

[19] C. M. Fonseca y P. J. Fleming. "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization". En S. Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms, pg. 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[20] D. E. Goldberg. "Genetic Algorithms in Search, Optimization and Machine Learning". Addison- Wesley Publishing Company, Reading, Massachusetts, 1989.

[21] J. Horn y N. Nafpliotis. "Multiobjective Optimization using the Niched Pareto Genetic Algorithm". Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[22] K. Miettien. "Some methods for nonlinear multi-objetive optimization". En E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, y D. Corne, editores, First International Conference on Evolutionary Multi-Criterion Optimization. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[23] A. G. et al. "PVM:Paralell Virtual machine - A user's guide and Tutorial for Networked parallel Computing". M.I.T. press, Cambridge, MA, 1994.

[24] N. Srinivas y K. Deb. "Multiobjective optimization using nondominated sorting in genetic algorithms". Technical report, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India, 1993.

[25] D. A. van Veldhuizen, J. B. Zydallis, y G. B. Lamont. "Issues in Parallelizing Multiobjective Evolutionary Algorithms for Real World Applications". En Proceedings of the 17th ACM Symposium on Applied Computing, pg. 595–602, Madrid, Spain, 2002. ACM Press.

[26] D. A. van Veldhuizen. "Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations". PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, Mayo 1999.

[27] E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, y D. Cornde, editores. "Proceedings of the First International conference on EMOO, 2001", Berlin, Germany, Marzo 2001. Springer-Verlag.

[28] E. Zitzler, M. Laumanns, y L. Thiele. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm", in K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou and T. Fogarty (eds.) EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, pp. 95--100, Athens, Greece, 2002.

[29] E. Zitzler y L. Thiele. "An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach". Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, Mayo 1998.

[30] E. Zitzler, K. Deb, y L. Thiele. "Comparison of Multiobjective Evolutionary Algorithms on Test Functions of Different Difficulty". En A. S. Wu, editor, Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program, pg. 121–122, Orlando, Florida, Julio 1999.

JaDiMa: Arquitectura de Máquina Virtual para la Construcción de Aplicaciones JAVA en Plataformas Grids

Yudith C. Cardinale, Eduardo A. Blanco y Jesús De Oliveira
Universidad Simón Bolívar, Dept. de Computación y T.I,
Caracas 1080-A, Venezuela, Apartado 89000
{yudith,eduardo}@ldc.usb.ve, jesus@bsc.co.ve

Abstract

This paper describes JaDiMa (*Java Distributed Machine*), a collaborative platform to construct high performance distributed Java applications. JaDiMa is a system that automatically manage the remote libraries used in a Java application. JaDiMa takes the advantages of portability, modularity, object oriented model and flexibility of Java, while incorporates communication and security well known techniques. The result is a simple and efficient distributed environment upon which applications and data are easily shared and highly portable amongst heterogeneous platforms and multiple users. JaDiMa allows compilation and execution of Java applications which use distributed libraries, without the necessity to keeping them in the developer and user hosts. To illustrate the functionality and characteristics of JaDiMa, we show examples of constructing real applications with several levels of library package dependencies in distributed environments.

Keywords: Management of distributed libraries, Reusability, Compilation with distributed libraries, Distributed execution, Grids platforms, High performance Java applications.

Resumen

Este artículo describe JaDiMa (*Java Distributed Machine*), una plataforma colaborativa para la construcción de aplicaciones Java distribuidas de alto desempeño. JaDiMa es un sistema que se encarga de la gestión automatizada de las librerías remotas utilizadas por una aplicación Java. JaDiMa explota las ventajas de la portabilidad, modularidad, modelo orientado por objetos y la flexibilidad de Java e incorpora técnicas de comunicación y de seguridad bien conocidas. El resultado es un ambiente de computación distribuido sencillo y eficiente sobre el cual las aplicaciones y los datos son fácilmente compartibles y altamente portables entre plataformas heterógeneas y múltiples usuarios. JaDiMa permite la compilación y ejecución de aplicaciones Java que utilizan piezas de software distribuidas, sin necesidad de mantenerlas locales a las máquinas de los desarrolladores o usuarios finales. Para ilustrar la funcionalidad y características de JaDiMa, presentamos ejemplos de la construción de aplicaciones reales con varios niveles de dependencias de librerías disponibles en ambientes distribuidos.

Palabras claves: Administración de librerías distribuidas, Reutilización, Compilación con librerías distribuidas, Ejecución distribuida, Plataformas grids, Aplicaciones Java de alto desempeño.

1 INTRODUCCIÓN

Los recientes avances en el uso de recursos distribuidos para la ejecución de aplicaciones de alto desempeño ha incrementado la posibilidad de ambientes colaborativos en el que múltiples usuarios, geográficamente distantes, comparten datos, piezas de software, recursos de cómputo, e incluso dispositivos especializados [3, 1]. En esta dirección, hoy en día es común el uso de varias librerías de software desarrolladas por terceros para lograr en conjunto el objetivo global requerido por una aplicación. Siguiendo el principio de la reutilización, resulta más práctico que los desarrolladores deleguen funcionalidades específicas de la aplicación en piezas de software, extensamente probadas y desarrolladas exclusivamente para tal función, y se concentren en la resolución del objetivo general.

En este sentido, los desarrolladores deben buscar y obtener las librerías adecuadas y referenciarlas en su código de manera que el proceso de compilación sea exitoso. Además, el usuario final debe poseer las mismas

librerías (las mismas versiones usadas en la compilación) para que la aplicación se ejecute correctamente. Por lo general, las aplicaciones son empaquetadas junto con las librerías de las que dependen, lo que implica que las piezas de software reutilizadas deben permanecer locales al ambiente de compilación y de ejecución. En un ambiente distribuido, este enfoque presenta graves desventajas: i) desperdicio de espacio en disco, en el caso en que una misma librería es utilizada por varias aplicaciones o que sólo una pequeña porción de código de dicha librería es utilizada; ii) dificultad en el manejo y actualización de las versiones de las librerías, al dejar en manos de los desarrolladores y usuarios finales la responsabilidad de actualizar sus versiones locales con versiones mejoradas y más recientes; y iii) para el caso particular de desarrollo de aplicaciones para grid computacionales, idealmente, las librerías sólo son requeridas para la compilación local, dado que la ejecución del código siempre se va a realizar en alguno de los nodos remotos del grid computacional. El hecho de descargar las librerías localmente sólo para compilación es un desperdicio de espacio y tiempo para el desarrollador.

Los lenguajes de programación tradicionales, como `C++`, `Fortran` y `HPF`, han sido por excelencia la preferencia de los desarrolladores de aplicaciones científicas. Sin embargo, recientemente Java se presenta como un lenguaje atractivo para la programación de aplicaciones en general, por su modelo de objetos sencillo y claro. Particularmente para ambientes colaborativos heterogéneos y distribuidos, una plataforma de ejecución portable, tal como la máquina virtual Java, hace potencialmente más fácil la implementación de aplicaciones científicas. El soporte de multi-hilos y concurrencia, combinado con la portabilidad inherente de Java, es especialmente interesante para las aplicaciones científicas paralelas y distribuidas. Si bien Java no es un lenguaje muy apropiado en la actualidad para realizar cómputo de alto rendimiento, la comunidad científica internacional está realizando grandes esfuerzos para lograr obtener, con Java, un rendimiento similar al obtenido con lenguajes más tradicionales como Fortran o C++. De hecho, ya existen resultados que muestran que Java tiene la potencialidad de alcanzar el desempeño de tales lenguajes científicos tradicionales [4, 8]. El uso de Java como lenguaje de programación de aplicaciones científicas y de alto desempeño está siendo promovido a través del foro *JavaGrande* [7]. Este foro pretende canalizar los esfuerzos para mejorar los aspectos del lenguaje relacionados con la computación de alto desempeño, incluyendo compiladores y librerías.

Este artículo describe JaDiMa (*Java Distributed Machine*), una plataforma colaborativa para la construcción de aplicaciones Java distribuidas de alto desempeño. JaDiMa es un sistema que se encarga de la gestión automatizada de las librerías remotas utilizadas por una aplicación Java. JaDiMa explota las ventajas de la portabilidad, modularidad, modelo orientado por objetos y la flexibilidad de Java e incorpora técnicas de comunicación y de seguridad bien conocidas (como el protocolo SOAP y el uso de certificados X.509). El resultado es un ambiente de computación distribuido sencillo y eficiente sobre el cual las aplicaciones y los datos son fácilmente compartibles y altamente portables entre plataformas heterógeneas y múltiples usuarios, motivando así la reutilización y compartimiento de librerías entre programadores novatos y avanzados.

JaDiMa permite la compilación y ejecución de aplicaciones Java de alto desempeño que utilizan piezas de software distribuidas, sin necesidad de mantenerlas locales a las máquinas de los desarrolladores de las aplicaciones. En este sentido, JaDiMa permite al desarrollador utilizar, durante la compilación, una representación mucho más pequeña de las librerías de las que depende su aplicación, denominados *stubs*. Los *stubs* son generedos automáticamente por JaDiMa y sustituyen las librerías reales. En tiempo de ejecución se descargan las definiciones reales de las clases de estas librerías desde repositorios de clases bien conocidos, a medida que van siendo requeridas por la aplicación. El uso de una convención para numeración de versiones permite la actualización automática de las librerías usadas por una aplicación sin afectar su correcto funcionamiento.

Para ilustrar la funcionalidad y características de JaDiMa, presentamos ejemplos de la construción de aplicaciones reales con varios niveles de dependencias entre librerías disponibles en ambientes distribuidos.

El resto del artículo está organizado como sigue. La Sección 2 presenta el diseño general de la arquitectura de JaDiMa. En la Sección 3 se describen los esquemas de comunicación y seguridad implementados. Un estudio comparativo con trabajos relacionados es presentado en la Sección 4. La descripción de cómo funciona JaDiMa, es presentada en la Sección 5, a través de ejemplos de construcción de aplicacines reales. Finalmente, en la Sección 6 resumimos las conclusiones derivadas del trabajo y proponemos posibles extensiones.

2 DESCRIPCIÓN DE LA ARQUITECTURA DE JaDiMa

El diseño de JaDiMa está orientado a satisfacer las siguientes características:

- Facilidad de instalación, configuración y uso. Se busca que JaDiMa sea funcional sin necesidad de seguir pasos complicados o procesos de configuración extensos. No se requiere de conocimientos técnicos para su instalación y uso. Cualquier usuario con un mínimo conocimiento en informática puede beneficiarse de las ventajas de JaDiMa. Se proveen de interfaces gráficas para facilitar la interacción con JaDiMa.
- Flexibilidad, adaptabilidad y modularidad. El diseño por capas y las interfaces generales de todos los componentes de JaDiMa, permite la integración de diferentes mecanismos estándares y bien conocidos de comunicación, seguridad y acceso a datos.
- Independiente de la plataforma. JaDiMa está totalmente implementado en Java. Los sistemas basados en Java resultan fáciles de usar tanto para programadores como para administradores. Esto hace que JaDiMa pueda ser instalado en cualquier plataforma que pueda ejecutar una máquina virtual Java, sin necesidad de recompilación.
- Transparencia. Ni las aplicaciones ni las librerías deberán ser modificadas para beneficiarse de las funcionalidades de JaDiMa.
- Alto rendimiento y escalabilidad. Para asegurar un buen desempeño de las aplicaciones que se ejecutan en el ambiente de JaDiMa, se provee de mecanismos de control de caches y de pre-carga de clases que reducen el impacto de la transferencias de las clases durante la ejecución.
- Seguridad. El intercambio de código y datos entre clientes y repositorios de librerías debe ser seguro. JaDiMa garantiza que todo código descargado proviene de un repositorio confiable, y que todo cliente que descargue código de una librería en particular está autorizado para ello. Además se implementaron mecanismos de control de acceso a las librerías para satisfacer la privacidad entre diferentes grupos de usuarios, si fuese requerida. Para la seguridad durante la ejecución, el código es ejecutado en "una caja de arena" (*sandbox*), que protege el acceso a los datos de la aplicación.
- Accesibilidad. Los clientes deben tener acceso a una amplia gama de librerías sin importar el lugar físico donde se encuentren publicadas, ni su plataforma de operación.

En las secciones siguientes describimos en detalle cómo la arquitectura de JaDiMa satisface estos requerimientos.

2.1 Componentes

El proceso de desarrollo de aplicaciones propuesto por JaDiMa se define en base a la identificación de las siguientes tareas: la administración de las librerías, su publicación, la compilación y la ejecución de las aplicaciones que requieran estas librerías. Cada una de estas tareas está representada por un componente: el Repositorio, el Agente de Publicación, el Agente de Compilación y el Agente de Ejecución, respectivamente. La figura 1 presenta el esquema general de la arquitectura de JaDiMa.

2.1.1 Repositorio

Este componente es el encargado de la administración de las librerías remotas que serán usadas en los procesos de compilación y ejecución definidos en JaDiMa. Por cada librería publicada, el Repositorio mantendrá tres conjuntos de datos:

1. la implementación de las clases reales suministradas por el usuario publicador y que serán usadas en la fase de ejecución;
2. las versiones reducidas (*stubs*) de las librerías, que serán usadas sólo durante el proceso de desarrollo de las aplicaciones.
3. la documentación de la librería.

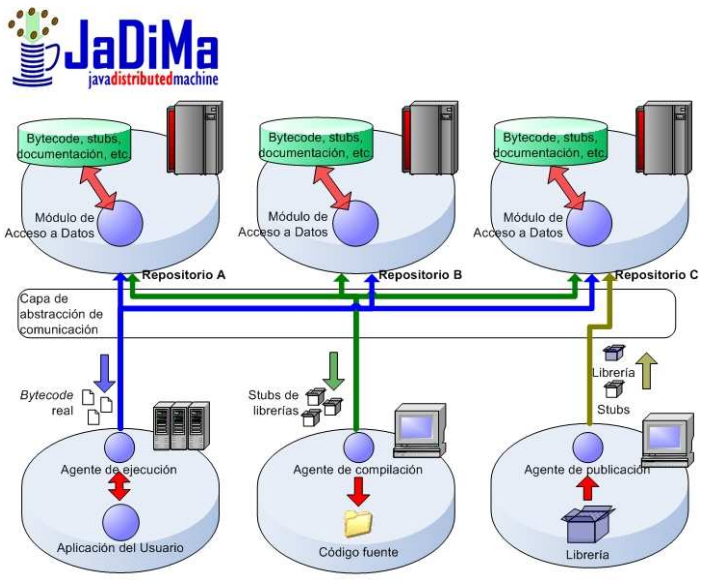


Figure 1: Arquitectura de JaDiMa

El Repositorio incluye tres elementos que permiten separar el medio físico de las operaciones de almacenamiento y recuperación de las librerías. El primero, denominado `Capa de Abstracción de Acceso a Datos`, sólo define las operaciones permitidas para la administración de las librerías (como publicar, consultar y obtener información) y no depende del medio físico de almacenamiento. El segundo, el `Módulo de Acceso a Datos`, es el que implementa las operaciones definidas de forma directa con el tercer elemento que es un `medio físico de almacenamiento`. La implementación de las operaciones se realiza a través de un manejador particular como MySQL, DB2, Oracle o un sistema de archivos tradicional. Este último es un ente externo a la definición de JaDiMa (Ver figura 2).

2.1.2 Agente de Publicación

La publicación de una librería en JaDiMa consiste de dos fases:

1. Generación de una representación reducida de la librería (*stubs*). Se genera un *stub* por cada clase de la librería: primero se crea un esqueleto de la clase, es decir, la definición de las firmas de todos sus métodos, en un archivo `.java`; luego estas definiciones son compiladas usando el Agente de Compilación de JaDiMa de manera de resolver las dependencias que tenga esta librería con otras previamente publicadas. La figura 3 refleja el proceso de generación de *stubs*. Los *stubs* serán usados por los desarrolladores en la fase de compilación de sus aplicaciones.
2. Transmisión de los datos (*stubs*, documentación y los paquetes de las librerías) desde el nodo de publicación hacia el Repositorio.

La instrucción que causa todo este proceso tiene la siguiente forma:

```
jdm_publish -b <library_name>.jar -v <major>.<minor>.<revision> -r <repository_name>
            -n <library_name>  -j <javadoc_zip_file>
```

donde:

-b <library_name>.jar, se refiere al paquete de la librería que se desea publicar.

-v <major>.<minor>.<revision>, es el número de versión asignado. La publicación de una librería requiere que su desarrollador le asigne un número de versión basado en el esquema de tres cifras [1], donde:

[1] Nuestro esquema de numeración de versiones propuesto puede ser adaptado de acuerdo al sistema de numeración de versiones de paquetes propuesto por Sun Microsystems [15].

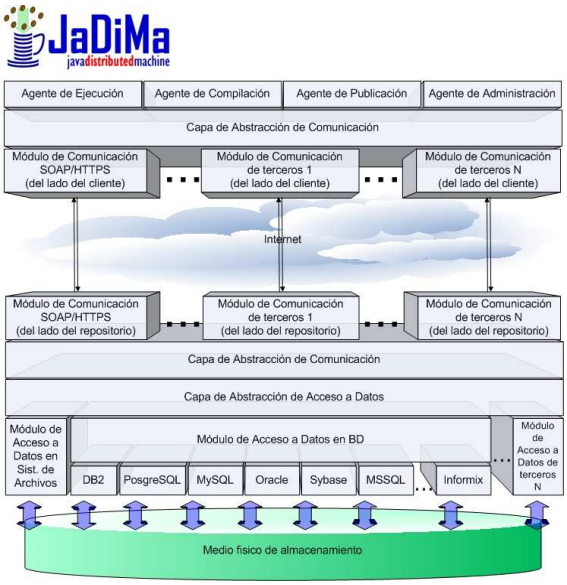


Figure 2: Vista lógica del Repositorio de JaDiMa

- `major`: representa cambios en funcionalidad extensos y en firma de los métodos
- `minor`: se refiere a cambios de funcionalidad menores, no hay cambios en firma de métodos.
- `revision`: no hay cambios de funcionalidad ni firma de métodos, corresponde a correcciones de errores en distribuciones anteriores de la librería.

-r <repository_name>, es el nombre del repositorio. En un archivo de configuración con formato `xml` se encuentra la información de contacto de los repositorios (por ejemplo, los nombres y los respectivos urls). Este archivo de configuración puede ser generado con ayuda de una interfaz gráfica.

-n <library_name>, es el nombre que identificará a la librería en el Repositorio.

-j <javadoc_zip_file>, indica el nombre del archivo `.zip` que contiene la documentación de la librería

2.1.3 Agente de Compilación

El Agente de Compilación es el encargado de obtener la información suministrada por el desarrollador sobre las dependencias de las aplicaciones y realizar solicitudes a los Repositorios especificados para obtener los *stubs* de las librerías. Con ayuda de una interfaz gráfica, el programador puede consultar la información sobre las diferentes librerías publicadas en los diferentes repositorios y seleccionar las de su interés. Luego que el desarrollador selecciona las librerías que requiere su aplicación, la interfaz gráfica genera automáticamente el archivo de metadatos que describe las dependencias de su proyecto. Durante el proceso de compilación, en ningún momento se solicita la descarga de las clases que fueron publicadas, en su lugar se obtienen los *stubs* que permiten que el proceso de compilación sea exitoso aun sin la presencia de las librerías reales. La instrucción de compilación tiene la siguiente forma:

```
jdmc -s <project's sources package root> -d <class files destination directory>
     <javac param1> .... <javac paramN>
```

donde:
-s <project's sources package root>, especifica el directorio donde se encuentran los archivos fuentes de las clases de la aplicación.

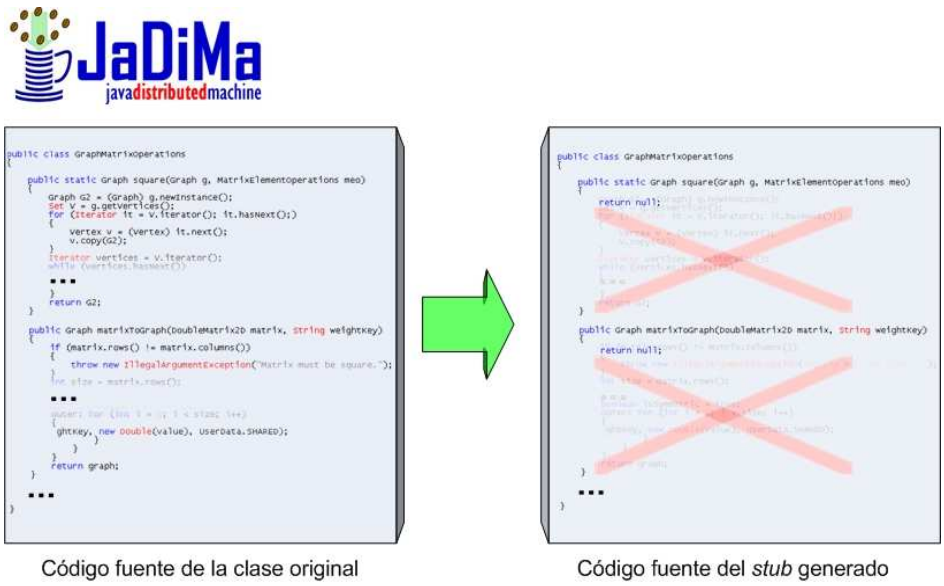


Figure 3: Generación de *stubs*

-d <class files destination directory>, especifica el directorio donde se alojaran los *stubs* y las clases compiladas.

<javac param1> .... <javac paramN>, pueden ser opciones propias de `javac`.

2.1.4 Agente de Ejecución

El Agente de Ejecución es el responsable de establecer un ambiente en el cual una aplicación pueda ejecutarse en forma transparente. Al solicitar la ejecución, el usuario posee una aplicación compilada usando JaDiMa (que contiene un *stub* de cada librería utilizada) y un archivo de dependencias que es usado para inicializar al `jdmClassLoader`. Cuando el código de la aplicación referencia alguna de las clases pertenecientes a las librerías remotas, el `jdmClassLoader` ubica y descarga, de los Repositorios, la definición real de la clase sustituyendo el *stub*, de manera de lograr una ejecución normal de la aplicación. Es importate destacar, que este esquema permite que solamente las clases utilizadas por la aplicación sean descargadas en el cliente; y además hace posible obtener versiones mejoradas de las librerías que no afecten la ejecución (según el esquema de numeración de versiones propuesto). La instrucción de ejecución es:

```
jdm -p <project's package root dir or jar> -m <main class name>
       <param1> ... <paramN>
```

donde:

-p <project's package root dir or jar>, es el directorio o `.jar` de la aplicación a ejecutar.

-m <main class name>, especifica el nombre de la clase principal.

<param1> ... <paramN>, son los parámetros de la aplicación o de la máquina virtual.

La figura 4 muestra el esquema gráfico del proceso de ejecución.

Precarga de clases y manejo de caché
Para lograr un mejor rendimiento durante la ejecución, implementamos un esquema de precarga de clases. El Agente de Ejecución mantiene información que representa una relación de asociación temporal entre las

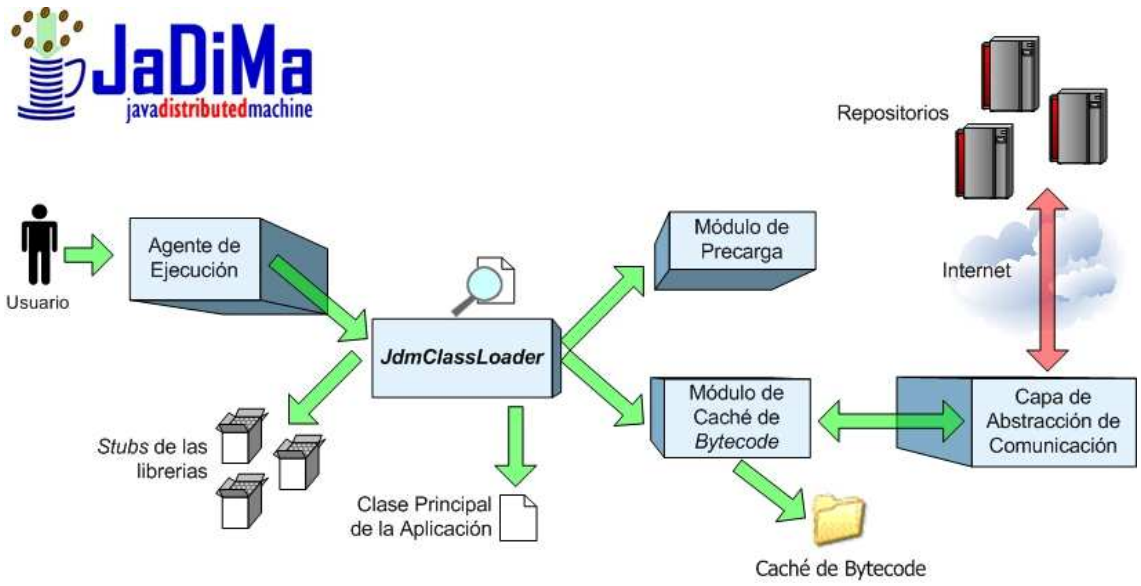


Figure 4: Modelo de ejecución en JaDiMa

clases que forman parte de una aplicación. Cada vez que una aplicación es ejecutada, esta información es actualizada de manera de lograr establecer los tiempos estimados en el que una clase es referenciada desde el comienzo de la ejecución de la aplicación. Esto permite crear *clusters* de clases que son referenciadas, por primera vez, dentro de un Δ de tiempo determinado. Así, cuando una clase es referenciada y no está presente en el nodo de ejecución, se solicitan todas las clases que pertenecen a su *cluster*. El módulo de precarga de clases es definido como un hilo (*thread*), de manera que puede ejecutarse concurrentemente con la aplicación, logrando solapar la transferencia de clases con el cómputo de la aplicación.

Por otro lado, el Agente de Ejecución administra la permanencia de las clases descargadas en un caché local. Esto significa que después que finaliza la ejecución de la aplicación, las clases de las librerías remotas permanecen en el Agente de Ejecución. De esta manera es posible reducir el retardo de comunicación en una siguiente ejecución. La implementación de políticas de reemplazo se planteó con un esquema de *plug-in* que refuerza la característica de adaptabilidad. Definimos una interfaz con métodos genéricos de acceso al caché, que pueden ser implementados con diferentes políticas de reemplazo. La política por defecto ofrecida actualmente es la de reemplazo del último menos recientemente utilizado (LRU).

3 ESQUEMAS DE COMUNICACIÓN Y SEGURIDAD EN JaDiMa

La comunicación entre los componentes de JaDiMa está basada en *plug-ins*, que permiten la implementación y uso de cualquier protocolo o mecanismo de comunicación. Los Componentes de JaDiMa interactúan con una `Capa de Abstracción de Comunicación` (ver figura 2), que es una interfaz bien definida que abstrae el modelo de comunicación de su implementación subyacente. Diferentes módulos que se adhieren a esta interfaz se encargan de implementar los mecanismos de comunicación efectivos entre los Agentes y los Repositorios para el intercambio de datos. En este sentido, la implementación de un nuevo esquema de comunicación consiste en el desarrollo de un módulo que implementa la interfaz de la `Capa de Abstracción de Comunicación`, usando el mecanismo de comunicación deseado.

El intercambio de información entre los Agentes y el Repositorio debe hacerse sobre ambientes seguros que provean mecanismos de autenticación de usuarios y de protección del canal de comunicacion. Los usuarios de JaDiMa deben registrarse con un rol determinado que describe las acciones que le son permitidas sobre el sistema, además debe poseer una credencial para efectos de la autenticación. Los roles actualmente manejados son: i) publicación, para realizar actividades de publicación, consulta y acceso a las librerías; ii) administración, para realizar actividades de administración sobre los Repositorios; iii) de acceso a librerías y versiones, para restringir el acceso a grupos de librerías. El Repositorio recibe, con cada solicitud, el rol del usuario (previamente autenticado en base al mecanismo de autenticación implementado a nivel del módulo

de comunicación) con el cual realiza las verificaciones de autorización.

El mecanismo de comunicación actualmente implementado en JaDiMa está basado en SOAP sobre HTTPS utilizando Apache Axis, pero puede ser sustituido por cualquier otro como CORBA, RMI, HTTP o sockets, gracias al diseño por capas de JaDiMa. La autenticación de usuarios se delega al servidor de aplicaciones Tomcat (*servlet container*) que utiliza el mecanismo de autenticación mutua con certificados digitales X.509. La asignación de un usuario autenticado a un rol de seguridad se realiza de acuerdo a los mecanismos configurados en el servidor de aplicaciones Tomcat, que mantiene información sobre los usuarios y sus roles. Esta información de seguridad puede ser administrada en bases de datos relacionales, archivos planos o directorios LDAP.

4 TRABAJOS RELACIONADOS

La reusabilidad de librerías especializadas para el desarrollo de aplicaciones científicas de alto desempeño, es un beneficio que se logra en ambientes colaborativos, como lo son los grid computacionales. Sin embargo, aún existen limitaciones en las herramientas que apoyan los procesos de compilación, despliegue, distribución y ejecución de dichas aplicaciones en sistemas grids. JaDiMa supera muchas de esas limitaciones y provee una arquitectura de máquina virtual sencilla y fácil de usar para la construcción de aplicaciones Java en plataformas distribuidas.

Apache Maven [2] y Krysalis Centipede [17], son proyectos que extienden las funcionalidades de herramientas tradicionales de compilación (como `ant` y `make`) para la gestión distribuida de las librerías. Al igual que JaDiMa, estos ambientes consideran que las librerías residen en servidores web que funcionan como repositorios, sin embargo, a partir del archivo de construcción (o archivo de metadatos), donde el programador describe las dependencias, se descargan las librerías en directorios locales que replican las estructuras de los repositorios. JaDiMa ofrece una interfaz gráfica que facilita la generación del archivo de metadatos en formato `xml`, a partir del cual sólo se descargan localmente los correspondientes *stubs* de las librerías. Además, dado que JaDiMa puede integrar diversos protocolos de comunicación, los repositorios no necesariamente deben ser servidores web. Con Apache Maven y Krysalis Centipede, la actualización de las versiones se realiza en tiempo de compilación, sólo si está explícito en el archivo de dependencias; en cambio en JaDiMa, se realiza en tiempo de ejecución. Basado en el sistema de numeración de versiones propuesto, el `jdmClassLoader` es capaz de descargar las últimas versiones o revisiones de las clases que son referenciadas sin afectar la ejecución. JaDiMa puede manejar seguridad de grano fino al permitir control de acceso a nivel de librerías y no de repositorios, en Apache Maven y Krysalis Centipede, la seguridad es manejada en la capa de comunicación. Ninguna de estas dos herramientas dan soporte a la ejecución.

DistAnt [5] y GridAnt [18] son sistemas que extienden el ambiente de construcción de archivos `ant` para proveer un procedimiento automático de despliegue o distribución de aplicaciones para su ejecución en plataformas distribuidas. El proceso de despliegue incluye transferencia de archivos, instalación y configuración en recursos de cómputo particulares para su posterior ejecución. DistAnt y GridAnt pueden ser combinados con Apache Maven para obtener los beneficios en el proceso de compilación, por supuesto heredando las desventajas, previamente mencionadas, que Apache Maven presenta. Al igual que JaDiMa, GridAnt usa certificados X.509 para ofrecer autorización y autenticación (con *single sign-on*). Los archivos de construcción de aplicaciones que consideran estos proyectos, están basados en el formato `xml` y deben ser producidos manualmente por los desarrolladores o usuarios, especificando los recursos de cómputo del grid en los cuales se copiaran las aplicaciones, se realizarán actividades de configuración e instalación y posteriormente su ejecución. En JaDiMa, el archivo de metadatos se genera automáticamente y el proceso de contactar los respectivos repositorios es transparente al programador. Para la ejecución, será la plataforma grid, quien seleccione los recursos de cómputo apropiados y el `jdmClassLoader` se encarga de localizar y descargar las clases reales de manera transparente. Tanto DistAnt como GridAnt, son herramientas orientadas a las actividades de los administradores para instalar software de manera remota, mientras que JaDiMa está orientado a apoyar las actividades de desarrollo de aplicaciones científicas.

Se han propuesto muchos proyectos que extienden la máquina vitual Java para facilitar la ejecución de aplicaciones distribuidas y paralelas en ambientes grids. Ejemplos de esos proyectos son Addistant [16], Unicorn [11], Javelin++ [9], JNLP [19], Bayanihan [12], HORB [13] y SUMA [6]. Ninguno de estos proyectos dan soporte a la compilación. Aun cuando la mayoría de estos trabajos tienen intereses diferentes a los de JaDiMa, son de particular importancia como herramientas complementarias en el ámbito de la ejecución.

Addistant permite adaptar una aplicación que se desarrolló para que se ejecutara en una única máquina

virtual, para que se ejecute en un ambiente distribuido (esto es distribución funcional). Este modelo de ejecución es justamente contrario al interés de JaDiMa, que consiste en ejecutar, en un recurso de computación, una aplicación cuyas partes pueden estar distribuidas. En este sentido, JaDiMa no modifica el modelo de ejecución original de la aplicación. En Addistant, los desarrolladores sólo especifican dónde se ejecutarán las instancias de cada clase en un archivo de "politicas" (*policy file*). A partir de este archivo, Addistant genera clases *proxies*, equivalentes a los *stubs* de JaDiMa, que serán las encargadas de realizar las referencias remotas. La aplicación es modificada automáticamente en tiempo de carga para sustituir las referencias a las clases que ya no son locales, por referencias a las respectivas clases *proxies*. JaDiMa hace la sustitución en tiempo de ejecución.

Unicorn, Javelin++, Bayanihan, HORB y SUMA proveen plataformas Java de ejecución distribuida que aprovechan el poder de cómputo en Internet. Los clientes proveen las aplicaciones junto con los datos a ser procesados. Las tareas son distriduidas en paralelo entre los recursos de cómputo ociosos conocidos por los respectivos servidores. El modelo de ejecución de JaDiMa, permite que sea fácilmente integrado a estas plataformas de ejecución, y así lograr un ambiente de soporte a la compilación y a la ejecución mucho más completo.

JNLP (*Java Network Launching Protocol*), es el proyecto que tiene más similitud con JaDiMa, en lo que respecta al modelo de ejecución. Al igual que JaDiMa, JNLP utiliza mecanismos relacionados con el *classloader* para cargar, durante la ejecución, las clases que componen una aplicación desde un servidor web. Sin embargo, en muchos aspectos no es transparente: las aplicaciones deben programarse con especificaciones particulares de paquetes propios de JNLP; el desarrollador debe especificar un archivo de metadatos con las propiedades de su aplicación y los módulos requeridos por la aplicación; si alguno de los módulos cambia, este archivo debe ser actualizado manualmente; requiere de la instalación de un *plug-in* en el navegador para reconocer el archivo de metadatos y comenzar la descarga parcial de los módulos de la aplicación. Es el único proyecto que considera pre-carga de clases pero es el desarrollador quien especifica cuáles clases se pueden cargar juntas, mientras que JaDiMa lo hace de manera transparente.

5 CASOS DE ESTUDIO: TIE y JUNG

Para ilustrar los conceptos que JaDiMa maneja y demostrar su funcionalidad, realizamos pruebas con aplicaciones reales de código fuente abierto con varios niveles de dependencias entre los paquetes usados. En esta sección particularmente nos referiremos a las experiencias con las aplicaciones TIE [14] (*Trainnable Information Extractor*) y JUNG [10] (*Java Universal Network/Graph*). TIE es un software entrenable para extracción de información y clasificación de texto. JUNG es una librería que provee un lenguaje extensible y común para el modelaje, análisis y visualización de datos que pueden ser representados como grafos o redes. Las dependencias de las librerías utilizadas por estas dos aplicaciones es mostrada en la tabla 1.

Las pruebas fueron realizadas en un escenario donde los componentes de JaDiMa se encontraban distribuidos en nodos en diferentes dominios y con diferentes ambientes y plataformas. Se utilizaron los recursos de los laboratorios de la Universidad Simón Bolívar (USB) y recursos externos a la USB, con las siguientes características:

- Pentium III de 800 MHz con 256 MB de RAM, 10 GB en Disco, en el Laboratorio Docente de Computación (LDC) de la USB.
- Pentium III duales de 800 MHz con 512 MB de RAM, 10 GB en Disco, en el Laboratorio de Sistemas Paralelos y Distribuidos (SPD) en la USB.
- AMD Athlon XP de 3.1 GHz, con 1 GB de RAM, 80 GB en Disco, y Dell dual de 800 MHz, 512 MB de RAM, 7 GB en Disco, en los nodos externos a la USB (BSC Consultores C.A).

Todos los nodos están conectados a Internet a través de enlaces de 256 Kbps en promedio. El escenario de pruebas fue el siguiente:

- Un Repositorio en el LDC (`LDC_REPOSITORY`), con Apache Tomcat 5.0.16 sobre Fedora Core 3, con un módulo de acceso a datos de tipo RDBMS, usando una base de datos MySQL localizada en otro nodo del mismo dominio.

Nombre de la Aplicación	Librerías usadas	Descripción
TIE	commons-beanutils	http://jakarta.apache.org/commons/beanutils/
	commons-collections	http://jakarta.apache.org/commons/collections/
	commons-configuration	http://jakarta.apache.org/commons/configuration/
	commons-discovery	http://jakarta.apache.org/commons/discovery/
	commons-lang	http://jakarta.apache.org/commons/lang/
	commons-logging	http://jakarta.apache.org/commons/logging/
	commons-math	http://jakarta.apache.org/commons/math/
	commons-pool	http://jakarta.apache.org/commons/pool/
	dom4j	http://www.dom4j.org/
	jtidy	http://jtidy.sourceforge.net/
	junit	http://www.junit.org/
	minorThird	http://minorthird.sourceforge.net/
	velocity	http://jakarta.apache.org/velocity/
minorThird	BeanShell	http://www.beanshell.org/
	MontyLingua	http://web.media.mit.edu/ hugo/montylingua/
	CRF	http://crf.sourceforge.net
	COLT	http://cern.ch/hoschek/colt/
	Gauss-Newton and Conjugate-Gradient optimization	http://billharlan.com/pub/code/inv/indexh.html
	JCommon	http://www.jfree.org/jcommon
	JFreeChart	http://www.jfree.org/jfreechart/index.html
	JUnit	http://www.junit.org
	JWF	http://sourceforge.net/projects/jwf/
	LBFS	http://crf.sourceforge.net/
	Libsvm	http://www.csie.ntu.edu.tw/ cjlin/libsvm/
	Log4J	http://jakarta.apache.org/log4j/docs/
	Trove	http://trove4j.sourceforge.net/javadocs/
JUNG	COLT	http://cern.ch/hoschek/colt/
	commons-configuration	http://jakarta.apache.org/commons/configuration/

Table 1: Dependencias de librerías

- Un Repositorio en un nodo de BSC Consultores C.A. (`BSC_REPOSITORY`), ejecutando JBOSS Application Server 4.0.0 sobre RedHat Linux 9, con un módulo de acceso a datos de tipo RDBMS usando una base de datos Microsoft SQLServer 2000, localizada en otro nodo del mismo dominio. El sistema de operación de estos nodos es Microsoft Windows 2000 Server.
- Un Repositorio en el Laboratorio SPD (`SPD_REPOSITORY`), con Apache Tomcat 5.5.9 sobre Scientific Linux CERN 3, con un módulo de acceso a datos a través de NFS.
- Nodos de publicación, compilación y ejecución en el Laboratorio SPD, con Scientific Linux CERN 3.
- Nodos de publicación, compilación y ejecución en el LDC, ejecutando Fedora Core 3.
- Un nodo de ejecución en un equipo portátil conectado a una red sin dominio, con Microsoft Windows 2000 Professional.

En todos los nodos se empleó la máquina virtual JAVA versión 1.5.0. La estrategia de pruebas consistió de los siguientes pasos:

1. Publicación de las librerías usadas por las aplicaciones (son las especificadas en la columna 2 de la tabla 1, excepto `minorThird`). Se publicaron diferentes versiones de estas librerías de manera aleatoria entre los tres repositorios, desde los diferentes nodos de publicación. El proceso de publicación implica la generación de los respectivos *stubs* y documentación, además del transporte de los datos (*stubs*, documentación y las propias librerías) desde el nodo de publicación hacia el repositorio especificado. Todo esto se realiza con una simple instrucción como la siguiente:

   ```
   jdm_publish -b ./commons-discovery.jar -v 1.6.0 -n commons-discovery -r BSC_REPOSITORY
   -j commons-discoverty-javadocs.zip
   ```

   Es importante que en el nodo de publicación esté presente el archivo de configuración que tiene la información de contacto de los repositorios (por ejemplo, los nombres y sus respectivos urls). Este archivo de configuración puede ser generado con ayuda de una interfaz gráfica.

2. Compilación de la librería `minorThird` en un nodo de compilación de SPD. Con ayuda de una interfaz gráfica, se seleccionan las diferentes librerías, publicadas en los diferentes repositorios, de las cuales depende `minorThird` y se genera el archivo de dependencias. Para compilar se usa:

```
jdmc -s /home/yudith/minorThird/src -d /home/yudith/minorThird/build -Xlint:none
```

En esta fase se descargan los respectivos *stubs* para que la compilación sea exitosa. Es necesario que en los nodos de compilación exista el archivo de configuración con la información de contacto de los repositorios. Así, el Agente de Compilación se conecta con los tres repositorios para resolver las dependencias.

3. Publicación de la versión compilada de `minorThird` en LDC_REPOSITORY. El proceso es parecido al explicado en el paso 1.

4. Compilación de la aplicación "TIE" en un nodo de compilación del LDC. El proceso es similar al explicado en el paso 2.

5. Ejecución de los "demos" de TIE en un nodo de ejecución externo a la USB, con la instrucción:

```
jdm -p /home/yudith/tie/build -m de.fu_berlin.ties.Main class-train -outdir=.
    -classifier=Winnow ../*.dsv
```

El Agente de Ejecución, primero descarga los *stubs* que no estén presentes en el directorio (los *stubs* pueden no estar presentes en el nodo ejecución porque la compilación se ejecutó en otro nodo o porque los *stubs* fueron reemplazados del caché), luego instancia el `jdmClassLoader` para cargar la clase principal. A medida que se van referenciado las demás clases, el `jdmClassLoader` se encarga de descargar las clases reales, tomando en cuenta el número de versión que se especifica en el respectivo *stub*.

6. Compilación de la aplicación JUNG en un nodo de compilación del LDC. En este caso el Agente de Compilación se conecta únicamente con los repositorios donde fueron publicadas las librerías `COLT` y `commons-configuration`.

7. Ejecución de los "demos" de JUNG desde el mismo nodo donde fue compilado.

La realización exitosa de estas pruebas, demuestra la funcionalidad de JaDiMa, e ilustra la facilidad de su uso.

6 CONCLUSIONES Y TRABAJO FUTURO

Con JaDiMa extendimos las potencialidades de la máquina virtual Java a un ambiente grid computacional. Como consecuencia, se provee a los desarrolladores y usuarios de aplicaciones científicas, de una plataforma colaborativa sencilla y eficiente, que permite fácil compartimiento de recursos de software y motiva la reusabilidad de código. JaDiMa se encarga de la gestión automatizada de las librerías remotas utilizadas por una aplicación Java. Los mecanismos de compilación y ejecución implementados permiten que las aplicaciones y los datos sean fácilmente compartibles y altamente portables entre plataformas heterógeneas y múltiples usuarios.

El diseño modular y por capas de JaDiMa, permite la fácil incorporación de mecanismos bien conocidos de comunicación, acceso a datos y seguridad.

Los resultados de nuestra exploración dan una visión positiva de la factibilidad y practicidad de ambientes de compilación y ejecución distribuidos con gestión automática de librerias remotas. Actualmente estamos trabajando en la integración de JaDiMa a suma, un grid computacional basado en Java. Con esa experiencia esperamos proveer un mecanismo que permita la fácil integración del ambiente JaDiMa a cualquier plataforma grid que provea ejecuciones remotas para aplicaciones Java.

References

[1] Ahmar Abbas. *Grid Computing: A Practical Guide to Technology and Applicactions*. Charles River Media, 2004.

[2] Apache Software Foundation. Apache Maven Project. http://maven.apache.org/.

[3] F. Berman, G. Fox, and A. Hey, editors. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, 2003.

[4] J. M. Bull, L. A. Smith, L. Pottage, and R. Freeman. Benchmarking Java against C and Fortran for scientific applications. In *Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande*, pages 97–105, Palo Alto, California, United States, 2001. ACM Press.

[5] W. Goscinski and D. Abramson. Distributed Ant: A System to Support Application Deployment in the Grid. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, volume 00, pages 436 – 443, Pittsburgh, PA, November 2004. IEEE Computer Society.

[6] E. Hernández, Y. Cardinale, C. Figueira, and A. Teruel. SUMA: A Scientific Metacomputer. In *Parallel Computing: Fundamentals and Applications. Proceedings of The International Conference ParCo99*, pages 566–573. Imperial College Press, 2000.

[7] Java Grande Forum. Java Grande Forum Report: Making Java work for high-end computing. Technical Report JGF-TR-1, Java Grande Forum Panel, 1998. Available at http://www.javagrande.org/sc98/-sc98grande.pdf.

[8] José Moreira. Closing the performance gap between Java and Fortran in technical computing. In *Proceedings of The Java for High Performance Computing Workshop, Europar 98*, 1998.

[9] Michael O. Neary, Sean P. Brydon, Paul Kmiec, Sami Rollins, and Peter Cappello. Javelin++: Scalability issues in global computing. In *Proceedings of the ACM 1999 conference on Java Grande*, pages 171–180, San Francisco, California, June 1999.

[10] Joshua O'Madadhain, Danyel Fisher, Tom Nelson, and Jens Krefeldt. JUNG: Java Universal Network/Graph Framework, 2003. http://jung.sourceforge.net/index.html.

[11] T. M. Ong, T. M. Lim, B. S. Lee, and C. K. Yeo. Unicorn: voluntary computing over Internet. *ACM SIGOPS Operating Systems Revie*, 36(2):36–51, April 2002.

[12] Luis F. G. Sarmenta and Satoshi Hirano. Bayanihan: building and studying web-based volunteer computing systems using java. *Future Generation Computer Systems*, 15(5-6):675–686, October 1999.

[13] Hirano Satoshi. HORB: Distributed Execution of Java Programs. In *Proceedings of the International Conference on Worldwide Computing and Its Applications*, pages 29–42, March 1997.

[14] Christian Siefkes. TIE: Trainable Information Extractor, 2003. http://www.inf.fu-berlin.de/inst/ag-db/software/tie/index.html.

[15] Inc. Sun Microsystems. Package version identification, 2002. http://java.sun.com/j2se/1.5.0/docs/guide/versioning/index.html.

[16] Michiaki Tatsubori, Toshiyuki Sasaki, Shigeru Chiba, and Kozo Itano. A Bytecode Translator for Distributed Execution of "Legacy" Java Software. In *Proceedings of the 15th European Conference on Object Oriented Programming (ECOOP 2001)*, volume 2072, Budapest, Hungary, June 2001. Springer-Verlag.

[17] The Krysalis Community Project. Krysalis centipede. http://krysalis.org/centipede/.

[18] G. von Laszewski, B. Alunkal, K. Amin, S. Hampton, and S. Nijsure. Gridant- client side grid workflow management with ant. Whitepaper, The Globus Alliance, July 2002. http://www-unix.globus.org/cog/projects/gridant/.

[19] John Zukowski. Deploying Software with JNLP and Java Web Start, August 2002. http://java.sun.com/developer/technicalArticles/Programming/jnlp/.

Estratégia para desenvolvimento de programas paralelos visando diminuir a intrusão causada por teste de software*

Leonardo Albernaz Amaral, Eduardo Augusto Bezerra

Hewlett-Packard/PUCRS – Centro de Pesquisa em Teste de Software (CPTS)
Faculdade de Informática (FACIN/PPGCC), PUCRS
Av. Ipiranga, 6681 – Prédio 30, 90619-900, Porto Alegre, RS, Brasil

{lamaral, eduardob}@inf.pucrs.br

Abstract

This work introduces a strategy to be used as part of a parallel programming methodology, aiming the reduction of the intrusion caused by software testing activities. In the proposed strategy, a discrete formalism (SAN) is used for the behavioural model representation of parallel applications. Test cases are created from this behavioural model to stimulate parallel programs, seeking out for inter-process communication errors. An important contribution of the proposed strategy is the generation and use of SAN-based test cases to reach coverage criteria, according to probabilities obtained from behavioural rates defined in the application model. The main idea behind the strategy is to come up with more deterministic test coverage schemes, mitigating the problem related to the large number of possible execution paths present in non-deterministic parallel programs.

Keywords: *parallel program testing and debugging, discrete formalism models, Stochastic Automata Networks.*

Resumo

Neste trabalho é apresentada uma estratégia para desenvolvimento de programas paralelos visando diminuir a intrusão causada pelas etapas de teste de software. A proposta baseia-se basicamente na idéia de utilizar um formalismo discreto (SAN) para a representação do modelo comportamental da aplicação e dessa forma criar casos de teste que exercitem a aplicação paralela na busca por falhas de comunicação entre processos. Uma contribuição importante dessa proposta é a utilização de casos de teste criados através de modelos SAN, o que possibilita critérios de cobertura baseados nas probabilidades obtidas a partir das taxas comportamentais determinadas no modelo da aplicação. Busca-se com essa estratégia, conseguir critérios de cobertura de teste mais determinísticos, diminuindo assim o problema do número excessivo de caminhos possíveis de execução geralmente encontrados em programas paralelos não-determinísticos.

Palavras chaves: *teste e depuração de programas paralelos, modelos de formalismo discreto, Redes de Autômatos Estocásticos.*

1. INTRODUÇÃO

O processamento paralelo pode ser considerado um método computacional onde se dividem grandes problemas em problemas menores que são processados simultaneamente por diferentes processadores, buscando-se alto desempenho na solução destes. A crescente aceitação do processamento paralelo deve-se principalmente ao aperfeiçoamento das redes de computadores, ao surgimento de *clusters* computacionais e também a computação distribuída [15].

Um *cluster* é basicamente uma coleção de máquinas distribuídas através de uma rede de alta velocidade, que serve como meio de comunicação entre essas máquinas. Seu potencial pode ser explorado adequadamente com a utilização do paradigma da troca de mensagens. Existem algumas opções disponíveis para dar suporte ao desenvolvimento de aplicações paralelas baseadas nesse paradigma como, por exemplo, o MPI (*Message Passing Interface*), que é implementado por meio de uma biblioteca de funções que provê suporte à comunicação entre processos.

Os *clusters* oferecem um enorme poder computacional e são usados para solucionar grandes desafios disseminados em diversas áreas estratégicas do conhecimento humano como: simulação e modelos de predição de

* Este trabalho foi desenvolvido em colaboração com a HP Brasil R&D.

desempenho (previsão do tempo, bioinformática, oceanografia), exploração de recursos de energia (explorações sísmicas e projetos de reservatórios), pesquisas médicas (tomografia computadorizada e estudos da engenharia genética), pesquisas militares (projetos de armas e simulações de ataques) efeitos visuais (filmes e animações), entre outros. O *software* para essa categoria de computadores deve ser desenvolvido de forma a fornecer resultados precisos e confiáveis, sendo o teste uma etapa bastante importante para o aumento dessa confiabilidade necessária.

Para o teste de *software* funcional, um programa seqüencial ou paralelo pode ser visto como um conjunto de estados e transições entre esses estados, as quais determinam o comportamento do sistema durante a sua execução. Para tal estratégia de teste, o comportamento do sistema pode ser abstraído através de formalismos para representação de modelos de uso e verificado através da consistência do comportamento observado durante a execução do programa, com o comportamento esperado, geralmente baseado na especificação do mesmo [51]. Em programas paralelos, as transições entre estados geralmente apresentam-se na forma de eventos de comunicação entre processos.

A descrição de aplicações por meio de modelos de uso é uma alternativa que vêm sendo utilizada pela comunidade científica, como uma estratégia para caracterizar o comportamento de um sistema, podendo dar mais detalhes às especificações do mesmo, principalmente quando associados a métodos estatísticos [52, 53, 6, 4].

Nesse trabalho é apresentada uma estratégia para desenvolvimento de programas paralelos visando diminuir a intrusão causada por teste de *software*. Essa proposta baseia-se em utilizar modelos de formalismo discreto (*SAN*) para a representação do modelo comportamental da aplicação paralela, e com isso mapear o modelo com o código da aplicação, determinando assim, o mapeamento dos eventos do modelo *SAN* com os eventos da aplicação. Dessa forma, através das estimativas probabilísticas de uso inferidas no modelo, cria-se casos de teste e *scripts* de teste funcional para a aplicação a ser testada e se define critérios de cobertura que permitam testar os programas paralelos utilizando-se além de técnicas de teste de *software*, algumas abordagens de depuração de programas paralelos.

As próximas Seções desse trabalho estão organizadas da seguinte maneira. As Seções 2 e 3 descrevem sucintamente os conceitos que serão utilizados e os principais trabalhos relacionados às ferramentas para criação, manipulação e análise de programas paralelos. Na Seção 4 é feito um detalhamento da proposta e uma discussão do andamento do trabalho. E por último, a Seção 5 apresenta algumas conclusões e trabalhos futuros.

2. TESTE DE SOFTWARE

Em conseqüência da crescente utilização de sistemas computacionais em praticamente todas as áreas do conhecimento humano, nas últimas décadas a Engenharia de *Software* evoluiu significativamente, procurando estabelecer técnicas, critérios, métodos e ferramentas para a produção de *software*. Tudo isso impulsionado por uma crescente demanda por qualidade e produtividade, tanto do ponto de vista do processo de produção, quanto dos produtos gerados.

Dentro desse contexto, a atividade de teste de *software* consiste basicamente em uma análise dinâmica do produto a ser testado, sendo considerada uma atividade relevante para a identificação e eliminação de falhas existentes em um sistema. Erros geralmente ocorrem devido a algum tipo de falha, seja esta humana ou física. Sistemas em estado errôneo são considerados sistemas defeituosos, visto que o processamento posterior a esse estado possivelmente cause defeitos operacionais, fazendo com que o sistema não se comporte conforme o esperado pela sua especificação [44, 7, 8]. O conjunto de informações oriundas da atividade de teste é significativo para as atividades de depuração, manutenção e estimativa de confiabilidade de *software* [19, 49].

Uma questão importante da atividade de teste, independentemente da fase, é a avaliação da qualidade de um determinado conjunto de casos de teste, visto que dependendo da aplicação, pode ser impraticável utilizar todo o domínio de dados de entrada para avaliar os aspectos funcionais e operacionais de um produto em teste. Assim, o objetivo do teste é utilizar casos de teste que tenham alta probabilidade de encontrar a maioria dos erros com um mínimo de tempo e esforço. Portanto, um teste bem sucedido é aquele que consegue determinar casos de teste para os quais o programa em teste falhe.

Basicamente existem três estratégias que podem ser aplicadas à fase de teste: a funcional *(Black Box*), a estrutural (G*lass Box*) e a baseada em métodos estatísticos. Na técnica funcional, os critérios e requisitos de teste são estabelecidos a partir de uma função de especificação do *software*, onde o objetivo é determinar se o programa satisfaz aos requisitos funcionais e não-funcionais que foram especificados [43]. Na técnica estrutural, os critérios e requisitos são derivados essencialmente a partir das características de uma particular implementação em teste, o que requer a inspeção do código fonte e a seleção de casos de teste que exercitem partes do código e não de sua especificação [9]. E na técnica baseada em métodos estatísticos, os critérios e requisitos de teste permitem o uso de inferências estatísticas para computar aspectos probabilísticos do processo de teste, tais como confiabilidade, tempo médio para falha e tempo médio entre falhas [12]. Observa-se também o estabelecimento de critérios de geração de casos de teste baseados em Máquinas de Estados Finito [16], Cadeias de Markov [53] e mais recentemente baseados em Redes de Autômatos Estocásticos [6].

2.1 Formalismo para Teste Estatístico

O teste estatístico com o passar dos anos passou a despertar grande interesse, pois viabiliza a superação de alguns pontos fracos de outras estratégias de teste. Ele geralmente está baseado em modelos de uso, os quais podem descrever possíveis comportamentos de um determinado *software*.

Os modelos de uso geralmente são representados por algum tipo de formalismo. Esses formalismos na maioria das vezes são baseados em grafos de estados com probabilidades de transição. O primeiro formalismo utilizado no teste estatístico foi Cadeias de *Markov*. Entretanto, existem estudos recentes em torno de um outro formalismo chamado Redes de Autômatos Estocásticos - *SAN* (*Stochastic Automata Networks*).

Na medida em que os modelos markovianos aumentam de tamanho e complexidade de seus componentes, as Cadeias de *Markov* apresentam limitações na capacidade de comportar tal crescimento. Nesse ponto, as Redes de Autômatos Estocásticos oferecem a capacidade necessária para tal representação.

Basicamente uma *SAN* é considerada um formalismo capaz de modelar um sistema em vários subsistemas, ou seja, um sistema composto de módulos quase independentes. Cada subsistema é representado por um autômato estocástico e por transições entre os estados deste autômato. Cada autômato é representado por um determinado número de estados, juntamente com regras ou funções de probabilidade que regem os movimentos de um estado para outro do autômato. O estado local de um autômato no tempo *t* é o estado que este autômato ocupa no tempo *t*. Já o estado global da rede de autômatos estocásticos é dado pelo estado local que cada um dos autômatos constituintes ocupa no tempo *t*. Os eventos em uma *SAN* podem ser eventos locais ou sincronizantes, e permitem mudar o estado global de uma *SAN*. Os eventos locais alteram o estado de apenas um autômato. Já os eventos sincronizantes, alteram simultaneamente mais de um estado local, ou seja, promovem a alteração de estados em mais de um autômato ao mesmo tempo.

O uso de Redes de Autômatos Estocásticos na representação de modelos de uso é exemplificado na Figura 1, onde é abstraído o comportamento de usuários na autenticação de um sistema. Sua estrutura possui os seguintes componentes:

- Autômatos (2): {*Login Automaton, Password Automaton*};
- Estados (6): {*Start, Menu, Password*} e {*PNotOk, Waiting, POK*};
- Eventos (5): {*ST, QT, S*} eventos sincronizantes e {*g, f*} eventos locais;

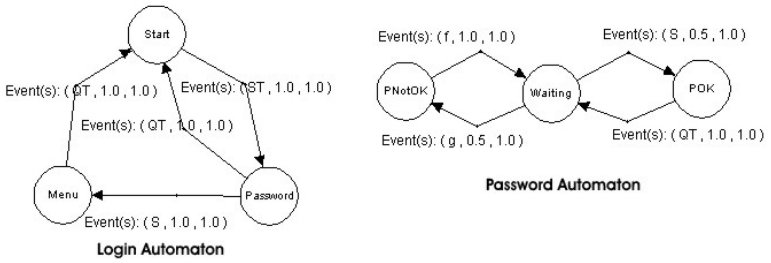


Fig. 1: Modelo *SAN* do sistema de *login*

O comportamento do sistema pode ser observado através de dois casos de teste para esse modelo, conforme a Figura 2. Nesses casos de teste são apresentados passo-a-passo os estados globais da rede e o evento que gerou a transição de estados. É possível notar que sempre que ocorre um evento sincronizante, os estados dos autômatos envolvidos nesse evento são alterados simultaneamente. Já quando ocorrem eventos locais, apenas estados locais ao evento são alterados.

Os casos de teste da Figura 2 foram criados em um ambiente integrado para geração de casos de teste e *scripts* para teste estatístico de *software* – STAGE [42]. É importante ressaltar que o ambiente STAGE foi desenvolvido no CPTS/PUCRS, em um projeto em colaboração com a HP Brasil. O formalismo utilizado na criação dos casos de teste foi *SAN*, e sofreu uma série de adaptações no modelo original. A principal contribuição desse sistema foi o uso desse formalismo para a representação dos modelos de uso e um modelo intermediário (*ISEM* – Modelo de Estado de Interface) para mapear a abstração do modelo de uso com a interface dos componentes do sistema. Dessa forma, a utilização de *SAN* permitiu uma representação modular de sistemas com comportamento complexo não-determinístico, minimizando a explosão de espaço de estados apresentado em Cadeias de *Markov*.

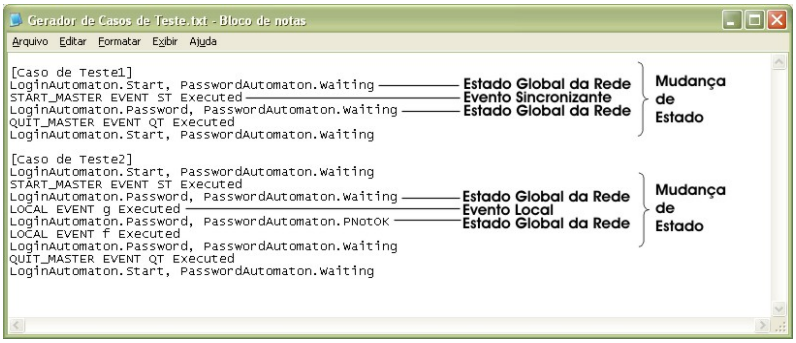


Fig. 2: Casos de teste do modelo *SAN*

No presente trabalho, busca-se estender as funcionalidades do STAGE para comportar a complexidade de programas paralelos, e com isso gerar automaticamente casos de teste e *scripts* para teste funcional de programas paralelos.

Além do trabalho apresentado em [42], outros trabalhos demonstram a utilização de modelos de uso e métodos estatísticos em etapas de teste de *software*, seja em programas seqüenciais ou em programas paralelos. Em [1] métodos estatísticos foram utilizados para analisar o comportamento de sistemas. A idéia do autor foi utilizar esses métodos para caracterizar as execuções de programas paralelos em modelos comportamentais e assim poder visualizar conjuntos específicos de dados inferidos estatisticamente através do modelo. Em [14], diversos modelos *SAN* foram criados e comparados com modelos de Cadeias de *Markov* em termos de números de estados, escalabilidade e capacidade de leitura desses modelos. Nessa comparação, *SAN* apresentou-se superior. Em [6], casos de teste foram gerados automaticamente através de modelos *SAN*. Já em [7], *SAN* é utilizada na construção de modelos de desempenho aplicados a programas paralelos, representando a comunicação entre processos paralelos com comportamentos síncronos e assíncronos.

Como a proposta desse trabalho está baseada na utilização de modelos de formalismo discreto para a representação comportamental de aplicações paralelas, fundamenta-se a utilização de Redes de Autômatos Estocásticos para essa abstração, devido a sua capacidade e flexibilidade para comportar a explosão de espaço de estados ocorridos na execução simultânea de processos paralelos. Entretanto, as etapas de teste tornam-se um pouco mais complicadas do que em programas seqüenciais, principalmente devido ao grande número de comunicação e sincronismo necessários. Dessa forma, torna-se indispensável a utilização de técnicas de depuração para a manipulação desses eventos de comunicação [30, 35, 32, 33].

Na próxima Seção, alguns conceitos relacionados a teste e depuração serão abordados, assim como os principais trabalhos que utilizam essa combinação de estratégias.

3. TESTE E DEPURAÇÃO DE PROGRAMAS PARALELOS

A princípio, um programa paralelo é uma coleção de diversos processos seqüenciais que são executados simultaneamente e se comunicam de alguma forma. Conseqüentemente, as dificuldades básicas encontradas em depuradores seqüenciais, são também evidentes para qualquer depurador paralelo [45]. Entretanto, além dessas dificuldades, os programas paralelos apresentam outros problemas. Em [30], o autor apresenta três justificativas para esses problemas: o aumento da complexidade dos programas paralelos, a quantidade de dados depurados e os efeitos anômalos adicionais.

O aumento da complexidade dos programas paralelos refere-se basicamente ao fato de que em programas paralelos se têm vários processos simultâneos, o que dificulta a estratégia tradicional utilizada em programas seqüenciais [18]. A segunda razão relaciona-se às dificuldades de se observar o que realmente interessa na depuração, evitando o acúmulo desnecessário de informações coletadas [46]. E a última razão diz respeito às dificuldades que ocorrem devido à concorrência e ao sincronismo entre os processos, o que não acontece nos programas seqüenciais [30]. Algumas dessas dificuldades são: *race-conditions* e *probe effects*. E outras se referem ao não-determinísmo e suas implicações nos programas paralelos como *irreproducibility effect* e *completeness problem*.

Um comportamento não-determinístico é caracterizado por violar o determinismo de execução, ou seja, múltiplas execuções de um mesmo programa resultando em diferentes seqüências de eventos sincronizantes e podendo produzir diferentes caminhos de execução [38]. Um exemplo de função de comunicação que introduz o não-

determinismo é a função de recebimento de mensagens *MPI_Recv (buffer, any_source)* da biblioteca de funções do *MPI*, onde o parâmetro *any_source* permite o recebimento aleatório de várias mensagens. Dessa forma não é possível determinar qual das mensagens será aceita primeiro. Isso vai depender da ordem de chegada dessas mensagens, o que caracteriza uma situação não-determinística, e com condição de corrida (*race-condition*) [39].

Existem dois problemas principais causados pelo não-determinismo, o *irreproducibility effect* e o *completeness problem*. O primeiro é caracterizado como sendo a incapacidade de realizar depuração cíclica (*cyclic debugging)*, ou seja, a incapacidade de reprodução de um erro encontrado na fase de teste, visto que o programa está sujeito a gerar a cada execução subseqüente, seqüências diferentes de eventos sincronizantes, ou diferentes caminhos de execução. Já o *completeness problem* está diretamente ligado a critérios de cobertura de teste, sendo um problema encontrado quando se tenta testar o comportamento de um programa paralelo em todos os caminhos possíveis de execução, o que dependendo do número de caminhos a serem testados pode se tornar impraticável.

Segundo [36], a solução para o *irreproducibility effect*, é a criação de um mecanismo que crie re-execuções equivalentes a uma execução anterior "observada". Esse mecanismo foi chamado de *record&replay*, e é composto por duas etapas. A primeira etapa é a fase de coleta (*record fase*), onde a ordem das mensagens são armazenadas em um histórico de execução (*tracefile*), o qual mantem informações sobre todos os eventos não-determinísticos. E a segunda etapa é a fase de re-execução (*replay*), onde os dados armazenados são utilizados para criar re-execuções equivalentes à execução "observada".

Para o tratamento do *completeness problem*, algumas técnicas também foram criadas como *controlled execution*, *event manipulation* e *artificial replay*. *Controlled execution* é uma abordagem proposta por [48], que oferece um método para a realização de teste automático. Baseada em uma técnica que descreve o comportamento desejado das comunicações entre os processos, o programa paralelo é executado em um comportamento forçado, evitando assim situações de condição de corrida. Uma outra abordagem dessa técnica foi proposta por [41], onde padrões de controle são aplicados para estabelecer as ordens entre os eventos de comunicação. Tais padrões são regras dinâmicas que definem a ordem de interação entre os processos.

Outros trabalhos tratam o *completeness problem* mais diretamente, conforme pode ser visto em [18, 27, 30]. Aqui os autores apresentam uma técnica chamada *event manipulation* e *artificial replay*. A idéia básica consiste em coletar informações referentes aos eventos de comunicação sincronizantes em uma execução inicial da aplicação, e assim identificar todas as condições de corrida existentes. Com isso, em uma segunda etapa, re-execuções artificiais são criadas através da manipulação das ordens dos eventos, permitindo que a aplicação tenha um outro comportamento. Os autores afirmam que conseguem investigar facilmente diferentes execuções para um mesmo conjunto de entradas. Além disso, todas as combinações possíveis de execução do programa podem ser criadas através da manipulação dos eventos, caso todas as combinações das mensagens de recebimento forem testadas. Inicialmente esse processo era manual [18], mas em [30] ele foi automatizado sem a necessidade de interação de usuários.

Algumas outras técnicas foram propostas para solucionar os problemas do não-determinismo [28, 25], mas a grande maioria está baseada em *record&replay.* Como o processo de coleta e armazenamento (*log*) das informações do programa paralelo é feito com a utilização de monitores, um monitor é basicamente um código extra (sonda) adicionado ao código fonte do programa, visando coletar informações. Essa estratégia também é conhecida como "observar um programa" [25]. O uso de monitores introduz o *probe effect*, ou *overhead* causado pela intrusão do processo de monitoração. Isso significa que a coleta de informações em tempo de execução pode influenciar o resultado do programa, seja na alteração dos tempos ou na ordem dos eventos.

Na próxima Seção um resumo das principais ferramentas existentes para a construção e análise de programas paralelos será apresentado. Inicialmente um modelo genérico para a classificação dessas ferramentas será mostrado. Logo após, estas ferramentas serão classificadas de acordo com suas principais características.

3.1 Modelo e arquitetura de ferramentas paralelas

Segundo [23], qualquer ferramenta de análise de programas paralelos consiste em dois itens principais, um componente de observação (monitoração) e um componente de análise. Outra característica importante é como e quando esses componentes interagem com a aplicação, sendo classificados em dois grupos: *on-line* e *off-line*. Quando a interação entre os componentes ocorre de maneira *on-line*, os dados do monitor são transportados durante a execução do programa para o componente de análise. Já no caso *off-line*, os dados necessários para análise são coletados pelo monitor enquanto o programa é executado, e salvos em um histórico de execução (*tracefile*) para serem analisados somente após o término da execução do programa.

Ambas as abordagens tem características diferentes. Inicialmente métodos *on-line* são mais flexíveis devido à capacidade de poder determinar e inspecionar cada estado durante a execução do programa. Por outro lado, os métodos *off-line* permitem algumas facilidades, principalmente porque ferramentas *on-line* apresentam apenas estados referentes

ao passado e ao presente, enquanto ferramentas *off-line* permitem uma análise completa dos estados da aplicação feita do início até o fim da execução do programa. Um exemplo dessa diferença pode ser percebido em técnicas para detecção de condições de corrida, onde métodos *on-line* são incapazes de identificar um conjunto completo de condições de corrida durante a execução do programa.

A maioria das ferramentas de monitoração são partes integrantes de ferramentas de análise, especialmente as ferramentas de análise *on-line*. Como exemplo é possível citar algumas ferramentas de depuração como *P2D2* [24], *PDBG* [12] e *PDT* [11] que utilizam o *GNU debugger – gdb* como depurador de baixo nível. Nesses casos, o monitor é completamente integrado no ambiente de depuração.

Quanto à capacidade de leitura dos históricos de execução (*logs* ou *tracefiles*) gerados pelas ferramentas de monitoração, a maioria dos formatos é em ASCII, o que oferece um alto grau de flexibilidade e leitura. Mas essa facilidade nem sempre é favorável, principalmente em casos em que a quantidade de dados monitorados é muito grande. Para esses casos, a solução encontrada foi criar *logs* em formatos binários, e em muitas vezes de maneira compactada, diminuindo dessa forma o seu tamanho e permitindo o armazenamento do comportamento dos programas de forma mais eficiente [47].

3.2 Resumo das principais ferramentas existentes

Baseado no modelo, nas características e nos problemas apresentados nas Seções anteriores, diversos trabalhos introduzem soluções inovadoras para tais problemas. Essas soluções correspondem ao estado da arte no contexto de ferramentas para criação, manipulação e análise de programas paralelos.

Em ambientes de teste e depuração existem trabalhos relacionados à descrição de ambientes completos que incluem a depuração como parte integral de suas estratégias de desenvolvimento. Alguns trabalhos discutem as vantagens dessa estratégia [2]. Em [35] é descrito uma combinação entre depuradores distribuídos – *DDBG* [12] com ferramentas de teste estrutural – *STEPS* [36]. Essa estratégia permite localizar erros através de análises simbólicas do código realizado com inspeções do código em execuções controladas. Uma estratégia parecida é oferecida por [54] em um ambiente integrado de ferramentas automatizadas para análise de programas que utilizam padrões de comunicação baseados em *PVM* (Tool-Set).

Outros exemplos de ambientes integrados podem ser vistos em *GRADE* (*Graphical Application Development Environment*) [26], e *MAD* (*Monitoring and Debugging Environment*) [31]. A idéia básica em *GRADE* é prover um suporte gráfico de alto nível para programação baseada em uma linguagem gráfica – *GRAPNEL* para ambientes *PVM*. Esse ambiente oferece módulos para a construção, execução, depuração, monitoração e visualização de programas paralelos baseados em troca de mensagens, podendo ser acessado pelos usuários através de uma interface gráfica. Como ferramenta de depuração, *GRADE* integra o *DDBG* como depurador distribuído.

Outro forte ambiente integrado que combina um conjunto de ferramentas para monitoração e análise de programas paralelos é o *MAD* [31]. De acordo com um modelo genérico para ferramentas de análise, os autores classificam os componentes de *MAD* em módulos de monitoração e análise. Diversas possibilidades são oferecidas em termos de monitoração. Uma implementação inicial chamada *EMU* (*Event Monitoring Utility*) gera históricos de execução para análise *off-line* em um formato de dados proprietário [32]. Uma das idéias principais é usar essa ferramenta para medir os efeitos causados na execução da aplicação com as técnicas de monitoração empregadas. Com isso, diversas estratégias foram implementadas para correção do *overhead* (intrusão) causado pelos monitores, objetivando remover a perturbação tanto no tempo de ocorrência dos eventos quanto em suas ordens.

Outra ferramenta pertencente ao *MAD*, e que se integra ao *EMU*, chama-se *PARASIT* (*Parallel Simulation Tool*), responsável por gerar execuções equivalentes em programas paralelos não-determinísticos. *NOPE* (*Nondeterministic Program Evaluator*) que representa um mecanismo completo para a técnica de *record&replay* é outra ferramenta integrada ao ambiente [30]. Ela apresenta uma característica adicional, a possibilidade de gerar manipulações automáticas de eventos com o objetivo de solucionar o *completeness problem* conforme apresentado nas Seções anteriores. Além dessas ferramentas, *MAD* apresenta ainda um módulo central – *ATEMPT* (*A Tool for Event Manipulation*), capaz de visualizar execuções paralelas utilizando diagramas baseados em técnicas de tempo e espaço (*space-time diagrams)* [33].

Para a representação gráfica do comportamento de programas paralelos, existem diversos trabalhos que descrevem abstrações de modelos comportamentais em diversos tipos de diagramas. Segundo [50], uma das melhores formas é através de Redes de Petri, o que pode ser aplicado em diferentes níveis de abstração de programas com características diferentes. Outro tipo de representação gráfica é o grafo de dependência (*dependency graph).* Ele é capaz de caracterizar visualmente dependências de dados e controle de operações executadas por um programa [5].

Existem outros tipos de representações gráficas, mas um dos mais usados é o diagrama de tempo e espaço, que foi introduzido por [34] para expressar ordem entre os eventos distribuídos. Esse diagrama é usado em muitas

ferramentas para depuração paralela e ajuste de desempenho [10]. Um bom exemplo é *Paragraph*, que possui além de diagramas de tempo e espaço, muitas outras formas de visualização [21]. Outra ferramenta que incorporou o diagrama de tempo e espaço foi *AIMS* [55]. *AIMS* é baseado em instrumentação em nível de código e provê portabilidade entre diferentes tipos de *hardware*. Uma ferramenta similar a *AIMS* é *XPVM* [17], e possui uma interface gráfica para análise de programas paralelos baseados em *PVM*, abstraindo graficamente tempos de computação, *overheads* de comunicação e tempos de espera, podendo ser visualizados tanto de maneira *on-line* quanto *off-line*. Uma comparação similar para programas *MPI* pode ser vista em *Upshot* [22] e *Vampir* [40], e mais recentemente em *Jumpshot4* [13]. Existem outros trabalhos com propostas semelhantes às citadas acima, entretanto uma revisão exaustiva foge ao objetivo do presente trabalho.

Essa Seção apresentou os trabalhos mais importantes e que servem de base para a definição da estratégia que está sendo proposta. Na próxima Seção, a estratégia será apresentada assim como o andamento do trabalho.

4. ESTRATÉGIA PROPOSTA

Nesse trabalho é apresentada uma estratégia cujo objetivo é possibilitar o desenvolvimento de programas paralelos mais testáveis, visando o uso de abordagens menos intrusivas nas etapas de teste. A estratégia está baseada em um processo de teste que define as etapas e técnicas utilizadas, conforme a Figura 3. Essas técnicas norteiam as contribuições mais importantes da estratégia e estão baseadas na geração de casos de teste e *scripts* para teste de *software* funcional, e na criação de um módulo de análise que teste os programas paralelos na busca por falhas na comunicação entre processos, conforme abordagens apresentadas nas Seções 2 e 3. Na Figura 4, o processo de teste é complementado com um ciclo de teste, onde suas etapas são especificadas. E na Figura 5, é definido um ambiente integrado baseado no processo de teste.

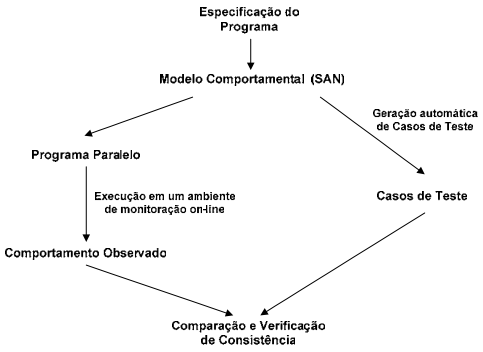


Fig. 3: Teste funcional baseado em modelos comportamentais

A geração dos casos de teste é feita automaticamente através dos modelos *SAN*, como sendo um possível ou esperado comportamento operacional da aplicação em teste. Com essa estratégia, busca-se criar através das estimativas probabilísticas dos modelos *SAN*, critérios de cobertura que possibilitem testar o sistema de maneira mais determinista. Conforme a Figura 4, o ciclo de teste é composto de diversas etapas, e algumas estão descritas a seguir:

- **Instrumentar programa:** processo de inserção de monitores no código da aplicação em teste conforme os estados determinados no modelo *SAN*. Com esses monitores no código, o módulo de análise é capaz de monitorar a execução da aplicação e identificar o conjunto de estados globais que caracterizam o comportamento observado;
- **Selecionar entradas:** etapa de leitura do *script* de teste que contem além dos casos de teste, parâmetros de execução que mapeiam os estados e autômatos do modelo com os estados e processos da aplicação;
- **Monitorar programa:** etapa de coleta do histórico de execução (*trace*) da aplicação em teste através da observação dos monitores inseridos no código. Essa etapa identifica os estados globais de execução e os eventos de comunicação;
- **Verificar a consistência de estados:** etapa que verifica se o estado global lido do modelo é atingido pela execução da aplicação.

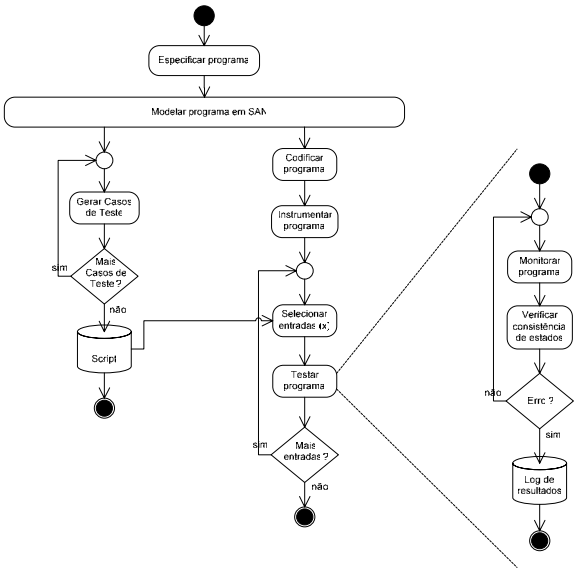


Fig. 4: Ciclo de teste do módulo de análise

Para a implementação do processo de teste, definiu-se um ambiente integrado para teste funcional de programas paralelos, conforme a Figura 5. O ambiente é composto por três módulos principais: módulo gerador de casos de teste e *scripts*; módulo de análise; e módulo de monitoração e execução, ambos desenvolvidos em JAVA. O módulo gerador de casos de teste e *scripts* estende os conceitos utilizados em [42], para gerar automaticamente casos de teste e *scripts* de teste funcional para aplicações paralelas.

O módulo de análise juntamente com o módulo de monitoração e execução, compõe o *engine* de teste e são os responsáveis pelo teste da aplicação. Conforme a Figura 5, e também às etapas especificadas no ciclo de teste, o módulo de análise recebe como estímulo (entrada), um *script* de teste contendo casos de teste e parâmetros de execução. A estrutura do *script* pode ser analisada através da Figura 7.

A interação entre os módulos e a aplicação em teste é feita de maneira *on-line*, onde os dados monitorados são transferidos para o módulo de análise em tempo de execução. Esses dados são coletados através de uma biblioteca de funções de monitoração, que é associada à execução do programa no instante da instrumentação do código da aplicação em teste. Com a integração desses módulos, é possível monitorar e analisar a execução da aplicação através dos estímulos oferecidos pelo *script* de teste, testando o comportamento da aplicação paralela na busca por situações de inconsistência entre os estados de execução e os estados do modelo. Assim, sempre que essas inconsistências forem identificadas, exceções são geradas em um histórico de resultados descrevendo as características envolvidas nos erros observados. Isso quer dizer que sempre que for identificado um estado errôneo, reporta-se o histórico de execução (*trace*) observado até esse estado atingido, tendo-se assim um conjunto de estados intermediários que ajudem na identificação da falha que causou o erro identificado.

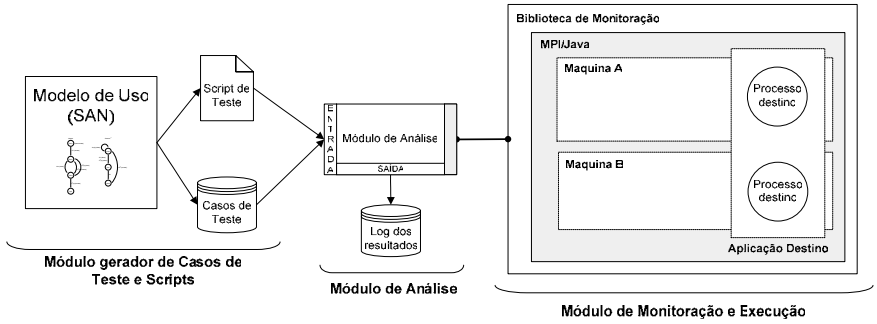


Fig. 5: Ambiente integrado para teste de programas paralelos

Como critérios inovadores, a estratégia apresenta a integração de técnicas consagradas pela comunidade científica em função de uma nova estratégia de teste, como o uso de monitores para a identificação dos estados de execução e métodos estatísticos para a geração automática de casos de teste. Um ponto forte da estratégia é a utilização de *SAN* para a representação de modelos comportamentais de programas paralelos, devido a sua capacidade de suportar a explosão de estados ocorridos devido à complexidade desse tipo de programa. Outro ponto forte é a utilização de abordagens de depuração na investigação das falhas encontradas no teste. Com essas abordagens, é possível identificar um conjunto intermediário de estados pertencentes à execução da aplicação até o estado de erro, e dessa forma poder investigar se esse erro foi causado por falha física ou humana.

4.1 Andamento do trabalho e principais funcionalidades

A estratégia proposta apresenta-se em fase de desenvolvimento. Estudos de caso estão sendo desenvolvidos para testar a funcionalidade da estratégia, e com a avaliação desses estudos concluída, será possível identificar os **aspectos positivos e negativos** do projeto.

A **metodologia** utilizada está diretamente vinculada com as **questões de pesquisa** envolvidas na definição do processo de teste e na implementação do ambiente de teste. Busca-se com essa metodologia, um melhor embasamento da real necessidade das atividades de teste no ciclo de desenvolvimento de aplicações paralelas. Espera-se também, levantar classes de erros mais cometidos por programadores, e com isso avaliar a combinação de estratégias de teste e depuração de programas paralelos aplicadas nesse trabalho.

Inicialmente havia-se definido uma técnica de teste baseada em execuções controladas [48, 41]. Mas devido à alta intrusão causada pelas etapas de monitoração exigidas por essa técnica, uma nova abordagem foi definida, onde basicamente o engine de teste estimula a aplicação com eventos externos e espera uma lista de variáveis de estado até o estado esperado. Com isso, é possível definir quais os estados a serem testados, e principalmente estabelecer uma abordagem um pouco menos intrusiva no comportamento da aplicação em teste. A definição do conjunto de estados a serem testados deve ser feita no *script* de teste. Para isso foi estipulado o uso de um arquivo no formato xml, conforme a Figura abaixo.

```
<script_engine naut="3">

    <processors>
        <procalias alias="master" id="1"/>
        <procalias alias="slave1" id="2"/>
        <procalias alias="slave2" id="3"/>
    </processors>

    <test_case id="1">
        <global_state>
            <local_state procid="1" stateid="Start_1"/>
            <local_state procid="2" stateid="Recv"/>
            <local_state procid="3" stateid="Recv"/>
        </global_state>
        <event name="ST_1" procid="1" type="START_MASTER"/>
        <global_state>
            <local_state procid="1" stateid="Start_2"/>
            <local_state procid="2" stateid="Ord"/>
            <local_state procid="3" stateid="Recv"/>
        </global_state>
        <event name="ST_2" procid="1" type="Master"/>
        <global_state>
            <local_state procid="1" stateid="Recv"/>
            <local_state procid="2" stateid="Ord"/>
            <local_state procid="3" stateid="Ord"/>
        </global_state>
        <event name="S3" procid="1" type="QUIT_MASTER"/>
        <final_state>
            <local_state procid="1" stateid="Quit"/>
            <local_state procid="2" stateid="Ord"/>
            <local_state procid="3" stateid="Ord"/>
        </final_state>
    </test_case>


</script_engine>
```

Fig. 7: Estrutura do *Script* de Teste.

Como o engine de teste monitora constantemente a execução da aplicação para a identificação dos estados globais, o *script* de teste deve conter apenas alguns estados a serem verificados na etapa de teste, visando com isso, diminuir a intrusão do monitor na execução da aplicação.

No *script* é definido o número de autômatos, um nome para identificação dos processos, e um conjunto de casos de teste gerados. Nesse conjunto de casos de teste, devem estar definidos os estados a serem testados, os quais

devem obedecer a seguinte ordem: **estado global** + **evento** + **estado global**, devendo o último estado ser definido como **estado final**. Alguns outros parâmetros estão sendo avaliados, como um critério de parada (*time-out*), um número máximo de tentativas de atingibilidade de um estado (*max_attempts*), e um controle do número máximo de mudanças de estados antes da ocorrência de um evento ou da atingibilidade de um estado respectivamente. Esses parâmetros servirão de guia para a identificação das prováveis causas dos erros identificados.

Para a avaliação dos estudos de caso, um modelo inicial de falha foi definido e está baseado em falhas de comunicação entre processos, sejam elas causadas por falhas em algum dispositivo de hardware (falhas físicas), ou falhas humanas cometidas na codificação do programa, o que é comum de acontecer devido à complexidade dos programas paralelos.

5. CONCLUSÕES E TRABALHOS FUTUROS

Nesse trabalho foi apresentada uma estratégia cujo objetivo é possibilitar o desenvolvimento de programas paralelos mais testáveis, visando o uso de abordagens menos intrusivas nas etapas de teste. Como motivação para o desenvolvimento desse trabalho, é possível citar a necessidade existente por métodos e técnicas para testes de aplicações paralelas, algo que é percebido através das inúmeras publicações existentes, conforme as Seções 2 e 3. Outra motivação é o crescente uso de clusters computacionais, os quais necessitam de *software*s confiáveis.

Embora a fase de avaliação dos estudos de caso esteja no começo, problemas devido ao não-determinismo já estão sendo enfrentados. Um dos principais é o *completeness problem*, que foi discutido na Seção 3. Esse problema dificulta o estabelecimento de um critério de cobertura para a geração dos casos de teste, visto que é inviável gerar casos de teste para todos os caminhos possíveis de execução.

Como solução, optou-se por utilizar *SAN* para a representação comportamental de programas paralelos, e gerar casos de teste baseados nas probabilidades do modelo. O problema neste caso, é que devido ao não-determinismo, é muito difícil fazer com que a aplicação em teste execute esse comportamento esperado várias vezes durante a sua execução, sem que haja algum método de manipulação de eventos, conforme apresentado na Seção 3.

Diante desse problema, está sendo avaliada a implementação de um módulo para a identificação de padrões, ou perfis de comportamento. Esse padrão é conseguido através de resultados probabilísticos identificados através da observação de sucessivas execuções de um programa [41, 3]. Com isso, são criados *ranks* com as probabilidades de ocorrência dos caminhos mais e menos executados, permitindo através desses resultados, uma realimentação do modelo com esses padrões observados. Com isso espera-se testar o programa de forma mais determinista.

Como trabalho futuro, pretende-se avaliar o uso desses padrões de comportamento como critério de cobertura nos testes que estão sendo realizados nos estudos de caso em andamento.

Referências

[1] Abrams, M., Ribler, R., Mathur, A., "***Two Performance Tool Design Issues and CHITRA´S Solution",*** 1996, Symposium on Parallel and Distributed Tools, Proceedings of the SIGMETRICS Symposium on Parallel and Distributed Tools, pp. 98-107, ACM Press, New York, NY, USA.

[2] Appelbe, W.F., McDowell, Ch.E., ***"Integrating Tools for Debugging and Developing Multitasking Programs"***, 1989, Special issue: Proceedings of the 1988 ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging, Vol. **24**(1): pp. 78-88, ACM Press New York, NY, USA.

[3] Bach, J., "***A Framework for Good Enough Testing***", 1998, Computer, Vol. **31**(10): pp. 124-126, IEEE Computer Society Press, Los Alamitos, CA, USA.

[4] Baldo, L., Brenner, L., Fernandes, L.G., Fernandes, P., Sales, A., "***Performance Models for Master/Slave Parallel Programs",*** **2004,** First Internationl Workshop on Practical Applications of Stochastic Modeling.

[5] Beguelin, A., Dongarra, J.J., Geist, A., Sunderam, V.S., ***"Visualization and Debugging in a Heterogeneous Environment"***. 1993, IEEE Computer, Vol. **26**(6): pp. 88-95, IEEE Computer Society Press, Los Alamitos, CA, USA.

[6] Bertolini, C., Farina, A. G., Fernandes, P., Oliveira, F. M., "***Test Case Generation Using Stochastic Automata Networks: Quantitative Analysis",*** 2004, Second International Conference on Software Engineering and Formal Methods (SEFM´04), pp. 251-260, IEEE Computer Society, Beijing, China.

[7] Bezerra, E. A., Vargas, F., Gough, M. P., ***"Improving reconfigurable systems reliability by combining periodical test and redundancy techniques: a case study"***, 2001, journal of Electronic Testing: theory and Applications, Vol.**17**(3): pp 701-711, Jetta, Norwell, Ma, Usa.

[8] Bezerrra, E. A., Jansch-Porto, I., "***Test procedure for faults detection in the transputer processor"***, 1997, Journal of solid State Devices and Circuits, Vol. **5**(1): pp 27-27, São Paulo.

[9] Boehm, B.W., ***"Software Engineering"***, *1976*, IEEE Transactions on Computer, Vol. **25**(12): pp. 1226-1241.

[10] Browne, J.C., Hyder, S.I., Dongarra, J., Moore, K., Newton, P., "***Visual Programming and Debugging for Parallel Computing"***, 1995, IEEE Parallel & Distributed Technology: System & Technology, Vol. **3**(1): pp. 75-83, IEEE Computer Society Press, Los Alamitos, CA, USA.

[11] Clemencon, C., Fritscher, J., Rühl, R., "***Visualization, Execution Control and Replay of Massively Parallel Programs within Annai's Debugging Tool"***. 1995, High Performance Computing Symposium, Montreal, Canada.

[12] Cunha, J.C., Lourenço, J.M., Vieira, J., Moscão, B., Pereira, D., ***"A Framework to Support Parallel and Distributed Debugging"***, 1998, Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking, Lecture Notes In Computer Science, Vol. 1401: pp. 708-717, Springer-Verlag, London, UK.

[13] Chan, A., Gropp, W., Lusk, E., "***Scalable Log Files for Parallel Program Trace Data*** *(DRAFT)"*, in *IL 60439*. 2000, Argonne National Laboratory: Argonne.

[14] Farina, A.G., Fernandes, P., Oliveira, F.M, ***"Representing Software Usage Models with Stochastic Automata Networks"***, 2002, Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, Vol. 27, ACM Press, New York, NY, USA.

[15] Foster, I.T., "***Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering****"*, 1995, p. 430, Addison-Wesley, Boston, MA, USA.

[16] Fujiwara, S., Bochmann, G. V., Khendek, F., Amalou, M., Ghedamsi, A., ***"Test Selection Based on Finite State Models"***, 1991, IEEE Transactions on Software Engineering, Vol. **17**(6): pp. 591-603, IEEE Press, Piscataway, NJ, USA.

[17] Geist, G.A., Kohl, J., Papadopoulos, P., "***Visualization, Debugging, and Performance in PVM"***, 1996, Proceedings. of Visualization and Debugging Workshop, in Debugging and Performance Tuning for Parallel Computing Systems, IEEE Computer Society Press, Los Alamitos, CA, USA.

[18] Grabner, S., Volkert, J., "***Debugging Distributed Memory Programs Using Communication Graph Manipulation****", 1996, International Symposium on High Performance Computing Systems*, Canada.

[19] Hartmann, J., Robson, D. J., "***Techniques for Selective Revalidation",*** 1990, IEEE Software, Vol. **7**(1): pp. 31-36, IEEE Computer Society Press, Los Alamitos, CA, USA.

[20] Hayes, A.H., Simmons, M.L., Brown, J.S., ***"Debugging and Performance Tuning for Parallel Computing Systems"***, 1996, p. 400, IEEE Computer Society Press, Los Alamitos, CA, USA.

[21] Heath, M.T., *"**Recent Developments and Case Studies in Performance Visualization using ParaGraph"***, 1993, Workshop on Performance Measurement and Visualization of Parallel Systems, pp. 175-200, Elsevier Science Publishers, B. V. Amsterdam, The Netherlands, The Netherlands.

[22] Herrarte, V., Lusk, E., "***Studying parallel program behavior with Upshot"***, in *Technical Report ANL-91/15*. 1991, Argonne National Laboratory, IL, USA.

[23] Hondroudakis, A., ***"Performance Analysis Tools for Parallel Programs",*** 1995, E.P. Computing Centre, The University of Edinburgh, (http://www.epcc.ed.ac.uk/epcc-tec/documents.html).

[24] Hood, R, "***The p2d2 Project: Building a Portable Distributed Debugger",*** 1996, ACM SIGMETRICS Symposium on Parallel and Distributed Tools, pp. 127-136, ACM Press, New York, NY, USA.

[25] Kacsuk, P, ***"Systematic Debugging of Parallel Programs Based on Collective Breakpoints"***, 1999, International Symposium on Software Engineering for Parallel and Distributed Systems. P. 83, IEEE Computer Society, Washington, DC, USA.

[26] Kacsuk, P., Cunha, J.C., Dozsa, G., Lourenco, J., Fadgyas, T., Antao, T., "***A Graphical Development and Debugging Environment for Parallel Programs", 1997,*** Journal of Parallel Computing, Distributed and Parallel Systems: Environments and Tools, Vol. **22**(13): pp. 1747-1770, Elsevier Science Press.

[27] Kranzlmüller, D., Grabner, S., and Volkert, J., "***Using Control and Data Flow Analysis for Race Evaluation"***, 1997, Proceedings of the Third International Euro-Par Conference on Parallel Processing, Eds. Lecture Notes In Computer Science, Vol. 1300, pp. 102-109, Springer-Verlag, London, UK.

[28] Krawczyk, H., Krysztop, B., Proficz, J, ***"Suitability of the Time Controlled Environment for Race Detection in Distributed Applications"***, 2000, Future Generation Computer Systems, Special Issue on Distributed and Parallel Systems, Vol. **16**(6): pp. 625-635, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands.

[29] Krawczyk, H., Wiszniewski, B, "***Interactive Testing Tool for Parallel Programs****"*, 1996, Software Engineering for Parallel and Distributed Systems, Chapman Hall, London, UK.

[30] Kranzlmüller, D., Volkert, J., "***Debugging Point-to-Point Communication in MPI and PVM***", 1998, Proceedings of the 5th European PVM/MPI Users Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes In Computer Science, Vol.1497: pp. 265-272, Springer-Verlag, London.

[31] Kranzlmuller, D., Grabner, S., Volkert, J., "***Debugging with the MAD environment"***, 1997, Parallel Computing, Special double issue on environment and tools for parallel scientific computing, Vol. 23 (1-2): pp. 199-217, Elsevier Science Publishers, Amsterdam, The Netherlands.

[32] Kranzlmüller, D., Grabner, S., J. Volkert, J., "***Monitoring Strategies for Hypercube Systems***.", 1996, Proceedings of the 4th Euromicro Workshop on Parallel and Distributed Processing (PDP ´96), pp. 486, IEEE Computer Society, Washington, DC,

USA.

[33] Kranzlmüller, D., Grabner, S., Volkert, J., ***"Message Passing Visualization with ATEMPT"***. 1995, Parallel Computing: State-ofthe-Art and Perspectives, *in: Proc. ParCo95, Conference on Parallel Computing*, Gent, Belgium.

[34] Lamport, L., "***Time, Clocks, and the Ordering of Events in a Distributed System***", 1978, Communications of the ACM, Vol. **21**(7): pp. 558-565, ACM Press, New York, NY, USA.

[35] Lourenco, J., Cunha, J.C. Krawczyk, H., Kuzora, P., Neyman, M., Wiszniewski, B., ***"An Integrated Testing and Debugging Environment for Parallel and Distributed Programs"***, 1997, Proceedings of the 23rd EUROMICRO Conference (EUROMICRO'97), pp. 291-298, IEEE Computer Society Press, Budapest, Hungary.

[36] Leu, E., Schiper, A., Zramdini, A. "***Execution Replay on Distributed Memory Architectures***", 1990, Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing, pp. 106-112.

[37] Murphy, G.C., Townsend, P. S., ***"Experiences With Cluster and Class Testing",*** 1994, Communications of the ACM, Vol. **37**(9): pp. 39-47, ACM Press, New York, NY, USA.

[38] Netzer, R.H.B., Miller, B.P, ***"Optimal Tracing and Replay for Debugging Message-Passing Parallel Program"***, 1992, Conference on High Performance Networking and Computing, Proceedings of the 1992 ACM/IEEE Conference on Supercomputing, pp. 502-511, IEEE Computer Society Press Los Alamitos, CA, USA.

[39] Netzer, R.H.B., Brennan, T.W., Damodaran-Kamal, S.K., ***"Debugging Race Conditions in Message-Passing Programs"***, 1996, Symposium on Parallel and Distributed Tools, Proceedings of the SIGMETRICS Symposium on Parallel and Distributed Tools, pp. 31-40, ACM Press, New York, NY, USA.

[40] Nagel, W.E., Arnold, A., Weber, M., Hoppe, H.-C., Solchenbach, K. ***"VAMPIR: Visualization and Analysis of MPI Resources"***. in *Supercomputer*. 1996.

[41] Oberhuber, M., ***"Elimination of Nondeterminacy for Testing and Debugging Parallel Programs"***, 1995, International Workshop on Automated and Algorithmic Debugging, Saint Malo, France.

[42] Oliveira, F.M., Copstein, B., Reginato, L. R.C, "***STAGE: an Integrated Environment for Statistical Test Script Generation***", 2004, Workshop de Testes e Tolerância a Falhas. . Gramado, RS, Brasil.

[43] Paul, J., "***Software Engineering - General Testing and Debugging Guidelines***", 1999, ttp://www.jodypaul.com/SWE/TD/TestDebug.html).

[44] Pradhan, D. K., "***Fault-Tolerant Computer System Design***", 1994, p. 550, Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[45] Rosenberg, J.B., ***"How Debuggers Work: Algorithms, Data Structures, and Architecture",*** 1996, p. 256, John Wiley & Sons, Inc., New York, NY, USA.

[46] Reed, D.A., Nickolayev, O.Y., Roth, P.C., ***"Real-Time Statistical Clustering for Event Trace Reduction"***, 1997, The International Journal of Supercomputer Applications and High Performance Computing, Vol. **11**(2): pp. 144-159.

[47] Ronsse, M.A., Levrouw, L.J., Bastiaens, K., "***Efficient Coding of Execution-Traces of Parallel Programs***", 1995, *Proceedings of the ProRISC / IEEE Benelux Workshop on Circuits, Systems and Signal Processing*, Utrecht.

[48] Tai, K.C., Carver, R.H., ***"Testing Distributed Programs"*** *(cap. 33).* in: Zomaya, A.Y., (Ed.), "Parallel and Distributed Computing Handbook", 1996, p. 1232, Mcgraw-Hill Computer Engineering Series, New York, NY.

[49] Valadan, G.S., ***"Trend´s in Reliability and Test Strategies,*** *1995*, IEEE Software, Vol. **12**(3).

[50] Vautherin, J., ***"Parallel Systems Specifications with Coloured Petri Nets and Algebraic Abstraction Data Types",*** 1987, Advances in Petri Nets, 7th European Workshop on Applications and Theory of Petri Nets, pp. 293-308, Springer-Verlag, Berlim.

[51] Wasserman, H., Blum, M., "***Software Reliability via Run-Time Result-Checking",*** 1997, Journal of the ACM (JACM), Vol. **44**(6): pp. 826-849, ACM Press, New York, NY, USA.

[52] Walton, G.H., Poore, J.H., Trammell, C.J., "***Statistical Testing of Software Based on a Usage Model",*** 1995, Software-Practice & Experience, Vol. **25**(1): pp. 97-108, John Wiley & Sons, Inc., New York, NY, USA.

[53] Whittaker, J.A., Thomason, M.G., "***A Markov Chain Model for Statistical Software Testing***", 1994, IEEE Transactions on Software Engineering on Special Section on the 15th Annual Internacional Conference on Software Engineering, Vol. **20**(10): pp. 812-824, IEEE Press , Piscataway, NJ, USA.

[54] Wismüller, R., Ludwig, T., Bode, A., Borgeest, R., Lamberts, S., Oberhuber, M., Röder, C., Stellner, G., "***The TOOL-SET Project: Towards an Integrated Tool Environment for Parallel Programs***", 1997, Proceedings of Second Sino-German Workshop on Advanced Parallel Processing Technologies - APPT'97, pp. 9-16, Verlag Dietmar Folbach, Koblenz, Germany.

[55] Yan, J.C., Jin, H.H., Schmidt, M.A., "***Performance Data Gathering and Representation from Fixed-Size Statistical Data",*** 1998, *Technical Report NAS-98-003, in: http://www.nas.nasa.gov/Research/Reports/Techreports/1998/nas-98-003.pdf,* NAS Systems Division, NASA Ames Research Center, Moffet Field, California, USA.

Reachability: a constrained path propagator implemented as a multi-agent system

Luis Quesada, Peter Van Roy, and Yves Deville
Université catholique de Louvain
Place Sainte Barbe, 2, B-1348 Louvain-la-Neuve, Belgium
{luque, pvr, yde}@info.ucl.ac.be

Abstract

Reachability is a propagator that implements a generalized reachability constraint on a directed graph g. Given a source node $source$ in g, we can identify three parts in the *Reachability* constraint: (1) the relation between each node of g and the set of nodes that it reaches, (2) the association of each pair of nodes $\langle source, i\rangle$ with its set of cut nodes, and (3) the association of each pair of nodes $\langle source, i\rangle$ with its set of bridges.

The effectiveness of our *Reachability* propagator has been shown by applying it to the Hamiltonian Path problem. The experimental evaluations that we have done in [10] show that it provides strong pruning, obtaining solutions with very little search. Furthermore, the experiments show that *Reachability* is also useful for defining a good distribution strategy and dealing with ordering constraints among mandatory nodes. These experimental results give evidence that *Reachability* is a useful primitive for solving constrained path problems over graphs.

In this paper we elaborate on the implementation of *Reachability*. *Reachability* has been implemented using a message passing approach on top of the multi-paradigm programming language Oz [7]. I.e., *Reachability* is a multi-agent system where agents send each other synchronous and asynchronous messages and their transition state functions rely on data flow and constraint programming primitives [14]. In the implementation, we profited from the Finite Integer Set module provided by the Mozart system, which implements Oz [5].

Keywords: constraint programming, filtering algorithms, message passing, concurrent programming.

1 Introduction

Constrained path problems have to do with finding paths in graphs subject to constraints. One way of constraining the graph is by enforcing reachability on nodes. For instance, it may be required that a node reaches a particular set of nodes by respecting some restrictions like visiting a particular set of nodes or edges and using less than a certain amount of resources. We have instances of this problem in Vehicle routing [9, 1, 6] and Bioinformatics [3].

An approach to solve this problem is by using Concurrent Constraint Programming (CCP) [13, 8]. In CCP, we solve the problem by interleaving two processes: propagation and distribution. In Propagation, we are interested in filtering the domains of a set of finite domain variables according to the semantics of the constraints that have to be respected. In Distribution, we are interested in specifying which alternative should be selected when searching for the solution.

In [10], we present *Reachability* as a propagator that is suitable for solving Hamiltonian Path with optional nodes. Given a directed graph g, a source node $source$ and a destination node $dest$, the Hamiltonian Path problem [2] consists in finding a path going from $source$ to $dest$ visiting every node of g once. In Hamiltonian Path with optional nodes, we are forced to visit only a specific subset of the nodes (instead of visiting all the nodes). We also show how a standard approach for dealing with this kind of problem, which is based on the use of *AllDiff* [11] and *NoCycle* [1], can be radically enhanced by using *Reachability*.

This paper is organized as follows: In section 2 we present the Reachability Constraint and a subset of its pruning rules. In section 3, we show how we can use a concurrent constraint language for defining propagators. In section 4, after presenting the CP(Graph) framework and its role in the implementation of *Reachability*, we show the implementation of the pruning rules in Oz. In this section, we also show that the composition of propagation, which we call *Batch Propagation*, can be easily achieved when using a message passing approach. The implementation of *Batch*

Propagation is an important result since it plays an important role in the reduction of the computation time. This is because it minimizes the number of activations of expensive propagators.

2 The reachability propagator

2.1 Reachability constraint

The Reachability constraint is defined as follows:

$$Reachability(g, source, rn, cn, be) \equiv \forall_{i \in N}. \begin{array}{l} rn(i) = Reach(g, i) \wedge \\ cn(i) = CutNodes(g, source, i) \wedge \\ be(i) = Bridges(g, source, i) \end{array} \quad (1)$$

Where:

- g is a graph whose set of nodes is a subset of N.
- $source$ is a node of g.
- $rn(i)$ is the set of nodes that i reaches.
- $cn(i)$ is the set of nodes appearing in all paths going from $source$ to i.
- $be(i)$ is the set of edges appearing in all paths going from $source$ to i.
- *Reach*, *Paths*, *CutNodes* and *Bridges* are functions that can be formally defined as follows:

$$j \in Reach(g, i) \leftrightarrow \exists_p . p \in Paths(g, i, j) \quad (2)$$

$$p \in Paths(g, i, j) \leftrightarrow \begin{array}{l} p = \langle k_1, ..., k_h \rangle \in nodes(g)^h \wedge k_1 = i \wedge k_h = j \wedge \\ \forall_{1 \leq f < h} . \langle k_f, k_{f+1} \rangle \in edges(g) \end{array} \quad (3)$$

$$k \in CutNodes(g, i, j) \leftrightarrow \forall_{p \in Paths(g,i,j)} . k \in nodes(p) \quad (4)$$

$$e \in Bridges(g, i, j) \leftrightarrow \forall_{p \in Paths(g,i,j)} . e \in edges(p) \quad (5)$$

The above definition of *Reachability* implies the following properties which are crucial for the pruning that *Reachability* performs. These properties define relations between the functions rn, cn, be, nodes and edges. These relations can then be used for pruning, as we show in section 2.2.

1. If $\langle i, j \rangle$ is an edge of g, then i reaches j.

$$\forall_{\langle i,j \rangle \in edges(g)} . j \in rn(i) \quad (6)$$

2. If i reaches j, then i reaches all the nodes that j reaches.

$$\forall_{i,j,k \in N} . j \in rn(i) \wedge k \in rn(j) \rightarrow k \in rn(i) \quad (7)$$

3. If i reaches j and k is a cut node between i and j in g, then k is reached from i and k reaches j:

$$\forall_{i,j \in N} . i \in rn(source) \wedge j \in cn(i) \rightarrow j \in rn(source) \wedge i \in rn(j) \quad (8)$$

4. Reached nodes, cut nodes and bridges are nodes and edges of g:

$$\forall_{i \in N} . rn(i) \subseteq nodes(g) \quad (9) \qquad \forall_{i \in N} . cn(i) \subseteq nodes(g) \quad (10) \qquad \forall_{i \in N} . be(i) \subseteq edges(g) \quad (11)$$

2.2 Pruning rules

We implement the constraint in Equation 1 with the propagator

$$Reachability(G, Source, RN, CN, BE) \tag{12}$$

In this propagator we have that:

- G is a graph variable [3] whose upper bound ($max(G)$) is the greatest graph to which G can be instantiated, and lower bound ($min(G)$) is the smallest graph to which G can be instantiated. So, $i \in nodes(G)$ means $i \in nodes(min(G))$ and $i \notin nodes(G)$ means $i \notin nodes(max(G))$ (the same applies for edges). In what follows, $\{\langle N_1, E_1\rangle \# \langle N_2, E_2\rangle\}$ will denote a graph variable whose lower bound is $\langle N_1, E_1\rangle$ and upper bound is $\langle N_2, E_2\rangle$. I.e., if $g = \langle n, e\rangle$ is the graph that G approximates, then $N_1 \subseteq n \subseteq N_2$ and $E_1 \subseteq e \subseteq E_2$.
- $Source$ is an integer representing the source in the graph.
- $RN(i)$ is a Finite Integer Set (FS) [5] variable associated with the set of nodes that can be reached from node i. The upper bound of this variable ($max(RN(i))$) is the set of nodes that could be reached from node i (i.e., nodes that are not in the upper bound are nodes that are known to be unreachable from i). The lower bound ($min(RN(i))$) is the set of nodes that are known to be reachable from node i. In what follows $\{S_1 \# S_2\}$ will denote a FS variable whose lower bound is the set S_1 and upper bound is the set S_2.
- $CN(i)$ is a FS variable associated with the set of nodes that are included in every path going from $Source$ to i.
- $BE(i)$ is a FS variable associated with the set of edges that are included in every path going from $Source$ to i.

The definition of *Reachability* and its derived properties give place to a set of propagation rules. We show here the most representative ones. The others are given in [10]. A propagation rule is defined as $\frac{C}{A}$ where C is a condition and A is an action. If C is true, the pruning defined by A can be performed.

- From (6) $\forall_{\langle i,j\rangle \in edges(g)}.j \in rn(i)$ we obtain:

$$\frac{\langle i, j\rangle \in edges(min(G))}{j \in min(RN(i))} \tag{13}$$

- From (7) $\forall_{i,j,k \in N}.j \in rn(i) \wedge k \in rn(j) \rightarrow k \in rn(i)$ we obtain:

$$\frac{j \in min(RN(i)) \wedge k \in min(RN(j))}{k \in min(RN(i))} \tag{14}$$

- From (8)$\forall_{i,j \in N}.i \in rn(source) \wedge j \in cn(i) \rightarrow j \in rn(source) \wedge i \in rn(j)$ we obtain:

$$\frac{i \in min(RN(Source)) \wedge j \in min(CN(i))}{j \in min(RN(Source))} \tag{15}$$

$$\frac{i \in min(RN(Source)) \wedge j \in min(CN(i))}{i \in min(RN(j))} \tag{16}$$

- From (1)$\forall_{i \in N}.Reach(g, i) = rn(i)$ we obtain:

$$\frac{j \notin Reach(max(G), i)}{j \notin max(RN(i))} \tag{17}$$

- From (1)$\forall_{i \in rn(source)}.cn(i) = CutNodes(g, source, i)$ we obtain:

$$\frac{j \in CutNodes(max(G), Source, i)}{j \in min(CN(i))} \tag{18}$$

- From (1)$\forall_{i \in rn(source)}.be(i) = Bridges(g, source, i)$ we obtain:

$$\frac{e \in Bridges(max(G), Source, i)}{e \in min(BE(i))} \tag{19}$$

- From (9) $\forall_{i \in N}.rn(i) \subseteq nodes(g)$, (10)$\forall_{i \in N}.cn(i) \subseteq nodes(g)$ and (11)$\forall_{i \in N}.be(i) \subseteq edges(g)$ we obtain:

$$\frac{k \in min(RN(i))}{k \in nodes(min(G))} \quad (20) \qquad \frac{k \in min(CN(i))}{k \in nodes(min(G))} \quad (21) \qquad \frac{e \in min(BE(i))}{e \in edges(min(G))} \quad (22)$$

One effect of these propagation rules is that, if $Source$ reaches j, and there is only one path p from $Source$ to j, then p is in G. For instance, consider the example in Figure 1. Let us assume that the source node is 1 and that 1 reaches 4 (this is why 4 is in the lower bound of node 1's FS variable). Then, let us add the constraint that edge $\langle 1, 3\rangle$ is not in G. This constraint and the information we already had imply that the path from 1 to 4 passing through 2 is in G since there is not other way of reaching 4 from 1.

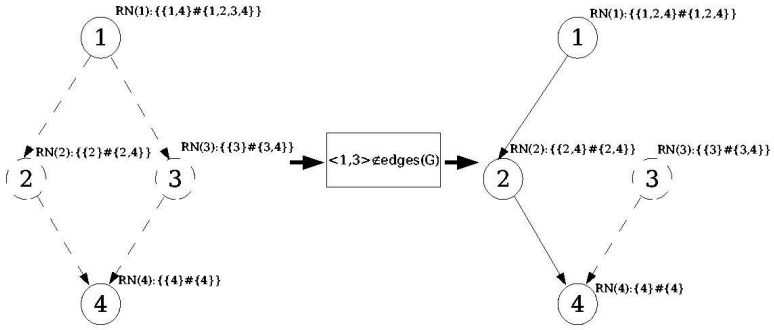


Figure 1: Pruning done by *Reachability* when the constraint $\langle 1, 3\rangle \notin G$ is added. Dashed nodes/edges are nodes/edges that may not be part of G (i.e., nodes/edges that are in the upper bound of G but not in the lower bound of G.

3 Defining propagators using a concurrent constraint language

We will define the Reachability propagator using a concurrent functional language, namely the declarative subset of Oz. This language is a concurrent constraint language in the sense of Saraswat [12]. For our purposes, it can be considered as a functional language that executes concurrently over a constraint store. The constraint store consists of a conjunction of primitive constraints. For example, in Figure 2 we observe that `Y` is the integer 42, `B` is a Finite Domain(FD) variable whose domain is $\{0, 1\}$, `S` is a FS variable whose lower and upper bounds are $\emptyset$ and $\{5\}$, `Msgs` is a list that is partially determined, and `Z` is a record with label `person` that has two fields: `age` whose value is the value of the variable `Y`, and `sex` whose value is `w`.

Information can only be added to the constraint store, by a "tell" operation, and never removed. Threads synchronize on information becoming available in the store, by an "ask" operation.

In our framework we distinguish three types of propagators:

- **Level 1**. These propagators are optimizations of propagators belonging to the two other levels that are provided by Mozart and implemented in C++. A propagator in this level can be considered as a thread that waits for information to become available, and then adds new information. For example, the propagator implementing the constraint `X=<:Y` reduces the upper bound of X to 10 when the constraint store knows that Y has upper bound 10.

- **Level 2**. A propagator in this level can be considered as a set of threads, each of which executes a recursive function that continuously waits for information to be added to the store, in order to add other information to the store. For instance, in Figure 4, `CreateCounter` creates a thread that reads its messages from the stream `S` and updates its state accordingly. This thread ceases to exist when reading the message `stop`. Notice that this thread computes a list containing the state values.

- **Level 3**. Propagators in this level can be seen as agents: active entities with which one can exchange messages (see chapter 5 of [14]). An agent is supposed to receive messages from different threads, so the order in

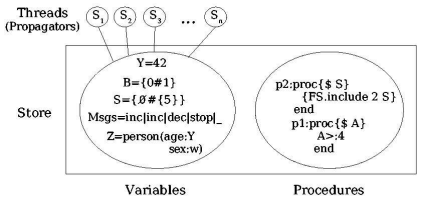


Figure 2: The Oz Execution Model (Declarative subset)

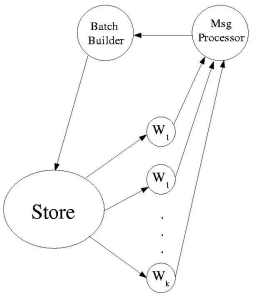


Figure 3: Architecture of a Graph variable propagator

```
proc {CreateCounter InitState S}
   fun {NextState state(val:Val output:Output) Msg}
      NewOutput
   in
      Output=Val|NewOutput
      case Msg of
         inc then state(val:Val+1 output:NewOutput)
      [] dec then state(val:Val-1 output:NewOutput)
      end
   end
   proc {ProcessMsgs State=state(val:Val output:Output) S}
      case S of stop|_ then Output=nil
      [] Msg|RestS then
         {ProcessMsgs {NextState State Msg} RestS}
      end
   end
in
   thread {ProcessMsgs InitState S} end
end
{CreateCounter state(val:0 output:Output) Msgs}
Msgs=inc|inc|dec|stop|_
```

Figure 4: A thread reading messages from a stream

which the agent receives the messages is completely indeterministic. This is why the agent is equipped with a communication channel (port) through which the messages are sent.

The global propagator of the graph variable that we are going to introduce in the next section is a level 3 propagator. The need of the communication channel comes from the fact that the order in which nodes/edges are introduced/excluded is not known a priori. Our solution is to have a thread per node/edge watching the insertion/exclusion of the node/edge. Once the node/edge is include/exclude the thread (which we call watcher) sends the corresponding message to the port. For instance, the following is the implementation of a node watcher. `Graph.N1.isIn` is 1/0 if `N1` is/is not in the graph. Once it is known that `N1` is/is not in the graph the watcher sends the message `includeNode(N1)`/`excludeNode(N1)` to the message processor.

```
thread
   if Graph.N1.isIn==1 then {Send MsgProcessor includeNode(N1)}
   else {Send MsgProcessor excludeNode(N1)} end
end
```

The interaction between the watchers and the message processor of the graph variable is shown in Figure 3. Notice that in this figure there is an additional component that we are going to introduce in section 4.4.

$S ::= S\ S$	Sequence
$\mid X = f(l_1 : Y_1 \ldots l_n : Y_n) \mid$	Value
$\mid X =$<number> $\mid X =$<atom>	
$\mid$ **local** $X_1 \ldots X_n$ **in** S **end** $\mid X = Y$	Variable
$\mid$ **proc** $\{\mathrm{X}\ Y_1 \ldots Y_n\}\ S$ **end** $\mid \{\mathrm{X}\ Y_1 \ldots Y_n\}$	Procedure
$\mid$ **if** X **then** S **else** S **end**	Conditional
$\mid$ **thread** S **end**	Thread

Table 1: The Oz declarative kernel language.

Each of the pruning rules of section 2.2 can be implemented in a straightforward way as a propagator using this computation model.

The declarative language we introduce here is based on procedures; semantically a procedure is similar to a process in a process calculus. This is because procedures can create threads and a thread can exist indefinitely as a running entity if it is implementing a propagator. We can still consider the language to be declarative, however, because it is confluent (see chapter 13 of [14]). Because of the monotonicity of the store, the concurrency executes in a restricted form that is deterministic and has no race conditions. This is clearly explained in chapter 4 of [14].

All Oz execution can be defined in terms of a kernel language whose semantics are given in chapter 13 of [14]. We will just refer to the declarative part of it.

Table 1 defines the abstract syntax of a statement S in the declarative subset of the Oz kernel language. Statement sequences are reduced sequentially inside a thread. All variables are logic variables, declared in an explicit scope defined by the local statement. Values (records, numbers, etc.) are introduced explicitly and can be equated to variables. Procedures are defined at run-time with the **`proc`** statement and referred to by a variable. Procedure applications block until the first argument references a procedure name. The **`if`** statement defines a conditional that blocks until its condition is **`true`** or **`false`** in the variable store. Threads are created explicitly with the **`thread`** statement. Each thread has a unique identifier that is used for thread-related operations.

In the following section, we are going to be using a bit of syntactic sugar to make programs easier to read. We will do so by considering that:

- **`proc`** `{...}` `...` **`in`** `...` **`end`** is equivalent to **`proc`** `{...}` **`local`** `...` **`in`** `...` **`end end`**.
- **`fun`** `{F V1 V2 ... Vn}` `<Stm>` `<Exp>` **`end`** is equivalent to **`proc`** `{F V1 V2 ... Vn R}` `...` `<Stm>` `R=<Exp>` **`end`**, where `<Exp>` is an expression representing a value and `<Stm>` is any statement.
- **`fun`** `{...}` `...` **`in`** `...` **`end`** is equivalent to **`fun`** `{...}` **`local`** `...` **`in`** `...` **`end end`**.

Procedures are values in Oz. This means that a variable may be bound to a procedure. In particular, we have that **`proc`** `{X V1...Vn}...` **`end`** is equivalent to `X=`**`proc`** `{$ V1...Vn}...` **`end`**, where the RHS is a procedure value.

4 Implementation of *Reachability*

4.1 CP(Graph)

CP(Graph) introduces a new computation domain focussed on graphs including a new type of variable, graph domain variables, as well as constraints over these variables and their propagators [3, 4]. CP(Graph) also introduces node variables and edge variables, and is integrated with the finite domain and finite set computation domain.

The kernel constraints of CP(Graph) are:

- $Nodes(G, SN)$: SN is the set of nodes of G.
- $Edges(G, SE)$: SE is the set of edges of G.
- $EdgeNode(E, N_1, N_2)$: the edge variable E is an edge from node N_1 to node N_2.

Consistency techniques have been developed, graph constraints have been built over the kernel constraints and global constraints have been proposed. $CP(Graph)$ has also been implemented in Oz [4].

4.2 Implementing CP(Graph) using message passing

In [10], we re-implemented part of CP(Graph) using a Message Passing approach, for implementing our *Reachability* propagator. We focussed on graph variables and provided the following implementation of the two first kernel constraints:

- `{G incN(N)}` results in $Nodes(G, SN) \wedge N \in SN$
- `{G exN(N)}` results in $Nodes(G, SN) \wedge N \notin SN$
- `{G incE(E)}` results in $Edges(G, SE) \wedge E \in SE$
- `{G exE(E)}` results in $Edges(G, SE) \wedge E \notin SE$
- `{G isN(N B)}` results in $Nodes(G, SN) \wedge (B = true \vee B = false) \wedge (N \in SN \leftrightarrow B = true)$
- `{G isE(E B)}` results in $Edges(G, SE) \wedge (B = true \vee B = false) \wedge (E \in SE \leftrightarrow B = true)$

Additionally, in our implementation, `{G stream($)}` is the stream that contains the messages associated with the constraints that have been imposed on `G`. So, if we have imposed the constraints:

```
{G incN(1)} {G incN(2)} {G exE(1#2)} {G incE(2#1)} {G exN(3)}
```

the partial value of `S` would be:

```
incN(1)|incN(2)|exE(1#2)|incE(2#1)|exN(3)|_
```

4.3 Pruning of *Reachability*

The skeleton of the implementation of *Reachability* is shown in Figure 5. In the implementation of *Reachability* there are two basic components: a set of already provided FS/FD propagators and a global (user defined) propagator. In this section, we will elaborate on the different propagators that constitute *Reachability* by referring to the pruning rules that they implement.

Notice that `CreateGlobalPropagator` creates an agent whose behavior is defined by the function `NextState`. The agent ceases to exist when encountering the message `determined` in the stream. `determined` signals the determination of the graph variable. G is determined when its lower bound is equal to its upper bound (i.e.,$min(G) = max(G)$). The determination of G implies that no message comes after `determined`.

4.3.1 Transitive closure of Reachability *(Rules 13 and 14)*

```
%% For every potential node I of G
/*1*/{FD.impl ({FS.card RN.I} >: 0) {G isN(I $)} 1}
/*2*/{FD.impl {G isN(I $)} {FS.reified.isIn I RN.I} 1}
```

Statement 1 imposes an implication between the cardinality of `RN.I` being greater than 0 and the presence of `I` in `G`. I.e., a node should be part of the graph in order to reach another one.

Statement 2 imposes an implication between the presence of `I` in `G` and `I` reaching itself. This is because every node of `G` reaches itself.

```
/*3*/Ss={G sucs($)}

%% For every potential pair of nodes <I,J> of G
/*4*/{FD.impl {FS.reified.isIn J Ss.I} {ReifiedSubSet RN.J RN.I} 1}
```

`Ss.I` is the set of successors of `I`. As these variables are already present in the implementation of graph variables, we simply make the corresponding associations between those variables and `Ss`(Statement 3).

Statement 4 imposes an implication between `J` being in `Ss.I` and `RN.J` being a subset of `RN.I`.

```
proc {Reachability G Source RN CN BE}
   ...
   proc {CreateGlobalPropagator G Source RN CN BE}
      fun {NextState state(graph:G) Msg}
         ...
      end
      proc {ProcessMsgs state(graph:G) Stream}
         case Stream of
            determined|_ then
            %% End of message processing
         [] Msg|RestStream then
            {ProcessMsgs {NextState state(graph:G) Msg} RestStream}
         end
      end
   in
      thread
         {ProcessMsgs state(graph:{MakeCompleteGraph NumNodes}) {G stream($)}}
      end
   end
in
   for I in 1..NumNodes do
      %% Unary propagators
      ...
      for J in 1..NumNodes do
         %% Binary propagators
         ...
      end
   end
   {CreateGlobalPropagator G Source RN CN BE}
end
```

Figure 5: Skeleton of Reachability

4.3.2 Pruning the upper bound of $RN(i)$ (Rule 17)

We first have to ensure that, for every `I` that is already known to belong to `G`, `RN.I` gets determined when `I` has no successors:

```
%% For every potential node I of G
/*5*/{FD.impl
      ({FS.card RN.I} >: 0)
      {FD.impl ({FS.card Ss.I} =: 0) ({FS.card RN.I} =: 1)}
      1}
```

We also have to ensure that `I` only reaches itself and the nodes that its successors reach. The following statement does that:

```
/*6*/local
        fun {Accumulate Sets J}
           if I\=J then S={FS.var.decl} in
              /*8*/{Select {G isInEdge(I#J $)} RN.J FS.value.empty S}
              S|Sets
           else Sets end
        end
        /*7*/SucSets={FoldL NodesIds Accumulate nil}
        /*9*/ReachedNodes={FS.unionN {FS.value.singl I}|SucSets}
     in
        /*10*/{Select ({FS.card RN.I} >: 0) ReachedNodes FS.value.empty RN.I}
```

```
   end
```

`SucSets`, defined in Statement 7, is bound to the sets of nodes reached by the successor. As we may not know a priori whether `J` is going to be successor of `I`, the corresponding set `S` is a set that is either the empty set (in case `J` is not a successor) or `RN.J`. This relation is imposed by the application of `Select`:

```
proc {Select Cond S1 S2 S3}
   {FS.subset S3 {FS.union S1 S2}}
   {FS.subset {FS.intersect S1 S2} S3}
   thread
      or Cond=1 S3=S1 [] Cond=0 S3=S2 end
   end
end
```

Depending on `Cond`, `Select` binds `S3` to `S1` or `S2`. Moreover, as `S3` is either `S1` or `S2`, `Select` constrains `S3` to have only the elements that `S1` and `S2` have and to include the elements that `S1` and `S2` have in common.

Statement 10 is the one that actually constrains `RN.I` to be the set containing `I` and the nodes reached by the successors of `I`. However, this is done on the condition that `I` is a node of `G` (i.e., `({FS.card RN.I} >: 0)`).

This is all what is needed for pruning a graph without cycles since the sets of reached nodes of the leaves get bound because of Statement 5, and this information is propagated to the corresponding predecessor because of Statement 10.

However, if `G` has cycles, the reached nodes sets do not get determined even if `G` is already determined. For instance, suppose that the lower and upper bound of `G` is `graph(1:[2] 2:[1])` and that the potential set of nodes is $\{1, 2, 3\}$. The propagators above mentioned will basically constrain `RN.1` to be equal to `RN.2` (and `RN.3` to be the empty set). Additionally, due to Statement 1 and 2, nodes 1 and 2 get into the lower bound of `RN.1` and `RN.2`. However, no propagator removes 3 from the upper bound of neither `RN.1` nor `RN.2`.

The upper bound of each reached nodes set is updated in the transition function of the global propagator of *Reachability*:

```
fun {NextState state(graph:G) Msg}
   case Msg of exE(N1#N2) then
      /*11*/NewG={RemoveEdge G N1#N2}
   in
      /*12*/{FS.subset RN.N1 {FS.value.make {DFS.reach N1 NewG}}}
      /*13*/{UpdateCutNodes CN Source NewG}
      /*14*/{UpdateBridges BE Source NewG}
      state(graph:NewG)
   else
      state(graph:G)
   end
end
```

The internal state of the global propagator is the upper bound of G. Each time an edge is removed, this upper bound is updated (Statement 11) and so are the upper bounds of the reached nodes sets affected (Statement 12). Notice that it is enough to update the reached nodes set of the origin of the edge removed (`N1`) since the rest will be done by Statement 10. Notice that `RN.N1` is updated by imposing that `RN.N1` is a subset of the nodes reached by `N1` in the upper bound `G`.

4.3.3 Discovering cut nodes

We have to start by keeping track of the cut nodes between the source and each other node (`CN.I`). As the set of cut nodes may change when an edge is removed, we update `CN.I` each time an edge removal takes place by invoking `UpdateCutNodes` (Statement 13). Notice that, in this statement, we are taking care of Rule 18 [1].

```
/*15*/{FD.impl {FS.reified.isIn I RN.Source} {ReifiedSubSet CN.I RN.Source} 1}

/*16*/{FD.impl {FS.reified.isIn J RN.I} {G isN(J $)} 1}
```

[1] We present the algorithms that we use for computing cut nodes and bridges in [10]. These algorithms are based on DFS [10].

In order to perform the pruning of rules 15 and 16. We impose an implication between `I` belonging to `RN.Source` and `CN.I` being a subset of `RN.Source` (Statement 15), and between `J` belonging to `RN.I` and `J` belonging to the nodes of `G` (Statement 16). In fact, this last statement also takes the pruning performed by rules 20 and 21 into account. An example illustrating the pruning performed by these statements is shown in Figure 6. In this example we impose the constraint that node 1 should reach node 9. As 5 is a cutnode between 1 and 9, 5 is included in `G` and forced to reach 9. Additionally, 1 is constrained to reach 5.

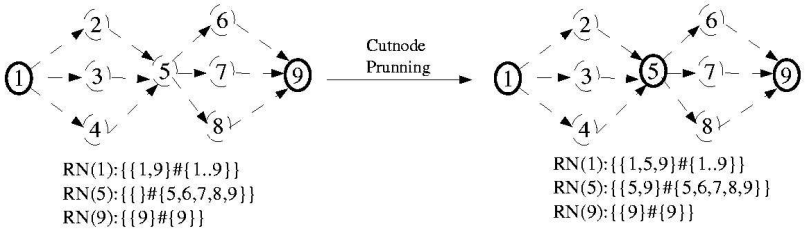


Figure 6: Discovering cut nodes

4.3.4 Discovering bridges

As in the previous case, `BE.I` is updated each time an edge removal takes place by invoking `UpdateBridges` (Statement 14).

```
/*17*/{FD.impl {FS.reified.isIn I RN.Source} {ReifiedEdgesInGraph BE.I G} 1}
```

We impose an implication between `I` belonging to `RN.Source` and the bridges between `Source` and `I` belonging to the edges of `G` (Statement 17). This statement covers the pruning of Rule 22 into account. An example illustrating the pruning performed by this statement is shown in Figure 7. In this example we impose the constraint that node 1 should reach node 5. This constraint is enough to determine the only path between 1 and 5.

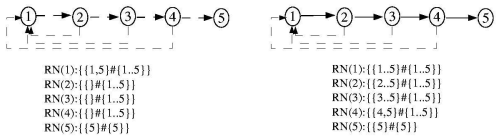


Figure 7: Discovering bridges

4.4 Batch propagation

In the previous implementation, we compute cut nodes and bridges each time an edge is removed. This certainly leads to a considerably amount of unnecessary computation since the set of cut nodes/bridges evolves monotonically. Another approach is to consider all the removals at once and make one computation of cut nodes and bridges per set of edges removed. This optimization can be implemented by adding a concurrent process to the implementation of graph variables. The task of this process is to batch together the messages according to their types (as shown in Figure 8). In this way, the transition function of the global propagator of *Reachability* will consider all the edges that have been removed at once:

```
fun {NextState state(graph:G)  batch(exE:Es ...)}
   if Es==nil then state(graph:G)
   else
```

Figure 8: Building batches

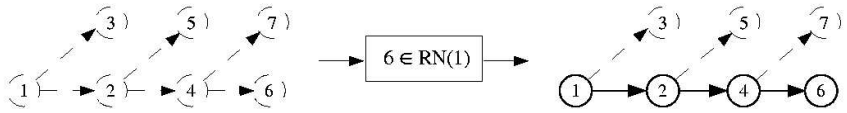


Figure 9: Simple Bridge Discovering

```
      NewG={RemoveEdges Es G}
   in
      {UpdateRNs Es NewG}
      {UpdateCutNodes CN Source NewG}
      {UpdateBridges BE Source NewG}
      state(graph:NewG)
   end
end
```

In fact, this transition function is very similar to the previous one. The only different thing is that `NewG` is considering all the nodes that have been removed.

Statement 6 is a cheap way of computing bridges when there is no cycle. Notice that, in the situation of Figure 9, the pruning performed by Statement 6 is enough for discovering the bridges between node 1 and node 6. However, the global propagator also discovers this information. The point in having this redundancy in propagation is that, thanks to the fact that the expensive propagator works on batches, there are cases where the expensive computation of bridges is not activated. Suppose, for instance, that discovering the bridge $\langle 2, 4\rangle$ raises a failure because 4 is not reached by 2. This failure is discovered by the cheap propagator and the expensive one is not activated.

5 Conclusion and future work

We presented the implementation of *Reachability*, which has been implemented using a message passing approach on top of the multi-paradigm programming language Oz [7]. We showed how the use of FS variables simplified the implementation of most of the rules.

In the implementation of *Reachability* we distinguished two basic components: a set of already provided FS/FD propagators and a global (user defined) propagator. We showed the global propagator as an agent that reads messages from a stream generated by the graph variable on which *Reachability* is applied.

We presented a cheap way of discovering bridges based on FS pruning. After introducing our implementation of Bath propagation using message passing, we explained why this can play an important role in the reduction of the computation time.

From our observations in [10], we infer that the suitability of *Reachability* is based on the strong pruning that it performs and the information that it provides for implementing smart distribution strategies. We also found that *Reachability* is appropriate for imposing dependencies on nodes. Certainly, we still have to see whether our conclusions apply to other types of graphs.

Our experiments in [10] also show that the appropriateness of *Reachability* is increased with the presence of optional nodes. This is basically because we are no longer able to apply the global *AllDiff* propagator on the successors of the nodes since we do not know a priori which nodes participate in the path. However, the complexity of the problem tends to increase with the number of optional nodes if they are uniformly distributed.

It is important to remark that both the computation of cut nodes and the computation of bridges play an essential role in the performance of *Reachability*. The reason is that each one is able to prune when the other can not. Notice that Figure 6 is a context where the computation of bridges cannot infer anything since there is no bridge. Similarly, Figure 7 represents a context where the computation of bridges discovers more information than the computation of cut nodes.

A drawback of our approach is that each time we compute cut nodes and bridges from scratch, so one of our next tasks is to overcome this limitation. I.e., given a graph g, how can we use the fact that the set of cut nodes between i and j is s for recomputing the set of cut nodes between i and j after the removal of some edges?.

The implementation of *Reachability* was suggested by a practical problem regarding mission planning in the context of an industrial project. Our future work will concentrate on making propagators like *Reachability* suitable for non-monotonic environments (i.e., environments where constraints can be removed). Instead of starting from scratch when such changes take place, what we want is to use the pruning previously performed in order to repair the pruning.

References

[1] Yves Caseau and Francois Laburthe. Solving small TSPs with constraints. In *International Conference on Logic Programming*, pages 316–330, 1997.

[2] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[3] G. Dooms, Y. Deville, and P. Dupont. Constrained path finding in biochemical networks. In *5èmes Journées Ouvertes Biologie Informatique Mathématiques*, 2004.

[4] G. Dooms, Y. Deville, and P. Dupont. CP(Graph):introducing a graph computation domain in constraint programming. Research Report INFO-2005-06, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 2005.

[5] Denys Duchier, Leif Kornstaedt, Martin Homik, Tobias Müller, Christian Schulte, and Peter Van Roy. *Finite Set Constraints*. December 1999. Available at *http://www.mozart-oz.org/*.

[6] F. Focacci, A. Lodi, and M. Milano. Solving tsp with time windows with constraints. In *CLP'99 International Conference on Logic Programming Proceedings*, 1999.

[7] Mozart Consortium. The Mozart Programming System, version 1.3.0, 2004. Available at *http://www.mozart-oz.org/*.

[8] Tobias Müller. *Constraint Propagation in Mozart*. Doctoral dissertation, Universität des Saarlandes, Naturwissenschaftlich-Technische Fakultät I, Fachrichtung Informatik, Saarbrücken, Germany, 2001.

[9] G. Pesant, M. Gendreau, J. Potvin, and J. Rousseau. An exact constraint logic programming algorithm for the travelling salesman with time windows, 1996.

[10] Luis Quesada, Peter Van Roy, and Yves Deville. The reachability propagator. Research Report INFO-2005-07, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 2005. Available at *http://www.info.ucl.ac.be/~luque/SPMN/paper.pdf*.

[11] Jean Charles Régin. A filtering algorithm for constraints of difference in csps. In *In Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 362–367, 1994.

[12] Vijay Saraswat. *Concurrent Constraint Programming*. The MIT Press, 1993.

[13] Christian Schulte. *Programming Constraint Services*. Doctoral dissertation, Universität des Saarlandes, Naturwissenschaftlich-Technische Fakultät I, Fachrichtung Informatik, Saarbrücken, Germany, 2000.

[14] P. Van Roy and S. Haridi. *Concepts, Techniques, and Models of Computer Programming*. The MIT Press, 2004.

Developing MAS Solutions with Gaia and AUML

Luca Cernuzzi[1,2], Franco Zambonelli[1]

1) DISMI – Università di Modena e Reggio Emilia, Italy
Via Allegri 13 – 42100 Reggio Emilia, Italy
2) DEI – Universidad Católica "Nuestra Señora de la Asunción", Paraguay
Campus Universitario – C.C. 1683 Asunción, Paraguay, Tel: +595-21-334650, Fax: +595-21-310072
e-mail: lcernuzz@uca.edu.py, franco.zambonelli@unimore.it

Abstract

A great number of Agent Oriented Software Engineering (AOSE) approaches and methodologies have been proposed in recent years, explicitly or implicitly, presenting different abstractions for the design and development of Multi-Agent Systems (MAS). This paper pays specific attention to the common abstractions used to model the complexity of MAS independently of the specific application. Moreover, it analyzes which of them are covered by Gaia integrated with AUML. Finally, as a case study, this paper presents an agent based solution modeled in Gaia with AUML and developed for the auditing process of the Central Bank in a developing country.

Keywords: Agent-based computing, Agent Modeling abstractions, Gaia, AUML.

1. INTRODUCTION

Agent-based computing represents an exciting new paradigm in the software engineering arena. Agents are being advocated as a next generation model for the engineering of complex, open and distributed systems. Yet, despite the increasing interest in these disciplines, some fundamental subjects, like the need for specific abstractions for modeling and developing MAS, have to be pointed out. In this sense, different authors have proposed different abstractions and different methodologies providing clear guidelines for the analysis, design, and development of MAS, and offering specific notations for each model [Ciancarini and Wooldridge, 2001], [Iglesias et al., 1999], [Jennings, 2000], [Jennings, 2001], [Cossentino and Zambonelli, 2004].
Agent-based computing includes a great variety of applications ranging from robot simulation to web distributed information systems. Because of this, it may be argued that these different types of applications may imply different abstractions. For example information searching or retrieval agents in internet may need mobility to move from one web site to another in order to accomplish their tasks, while intelligent agents claim for specific skills of reasoning.
However, independently of the application of any system in particular, our hypothesis is that some common abstractions can be identified for the agent-based computing paradigm.
Thus, a first question is: "is it possible to find convergence on some "must" abstractions for MAS independently of the specific application?". If so, it would be possible to verify the adequacy of the proposed AOSE approaches and methodologies.
For this reason, we are interested in analyzing if Gaia integrated with AUML is a good candidate for MAS design and if it adequately captures these common abstractions. To do so, we also propose a case study centered on the design and development of an agent based solution, modeled in Gaia integrated with AUML, developed for the auditing process of the Central Bank of Paraguay.
The rest of the paper is structured as follows. Section 2 introduces a proposal for common agent-based computing abstractions; a brief summary of Gaia and AUML; and an analysis of how Gaia+AUML covers the proposed common abstractions. Section 3 presents the case study of the auditing system for the Central Bank of Paraguay. Section 4 discusses the proposal comparing it with related works. Finally, section 5 concludes and sketches future works.

2. COMMON ABSTRACTIONS IN GAIA AND AUML

Different abstractions have been proposed to characterize MAS. Some of them are mainly dependent on the type of MAS that the designers take into consideration (e.g. MAS with mobility); while others are common abstractions that

are quite independent from the specific application. For generalization purposes, in this study we will mainly focus on this second group. Thus, among the common abstractions for MAS it is possible to identify:

- roles:
- activities with their corresponding goals
- agents:
- interaction with the environment:
- organizational aspects (structure and control):
- interaction between agents.

Currently, almost all traditional and emergent methodologies, as well as the engineering approaches proposed for agent-based systems design and construction, cover the mentioned abstractions. However, they do it in different ways, and some of them organize the process and the models in a more coherent framework than others. Some suggestions on this regard may arise from different interesting works on the evaluation of methodologies [Sturm and Shehory, 2003], [Cernuzzi and Rossi, 2002], [Hoa Dam and Winikoff, 2003], etc.

Considering this panorama, we are interested in analyzing how Gaia [Zambonelli et al., 2003], and its integration with AUML [Bauer et al., 2000; Odell et al. 2000], can be considered a good choice for designers. Therefore, in the next subsections we briefly introduce Gaia and AUML, and then analyze their adequacy in efficiently capturing the common abstractions for MAS.

2.1. Gaia in a Nutshell

Gaia [Zambonelli et al., 2003] focuses on the use of organizational abstractions to drive the analysis and design of MAS. Modeling both the macro (social) and the micro (agent internals) aspects of a MAS, Gaia devotes specific efforts to model the organizational structure and the organizational rules that govern the global behavior of the agents in the MAS organization. Gaia mainly covers the analysis and the architectural design phases.

The goal of the analysis phase in Gaia, which covers the requirements in term of functions and activities, is firstly to identify the loosely coupled sub-organizations that possibly compose the whole systems and then, to produce four basic abstract models: (i) the environmental model, to capture the characteristics of the MAS operational environment; (ii) a preliminary roles model, to capture the key task-oriented activities to be played in the MAS; (iii) a preliminary interactions model, to capture basic inter-dependencies between roles; and (iv) a set of organizational rules, expressing global constraints/directives that must underlie the MAS functioning.

The above analysis models are used as input to the architectural design phase. In particular, the architectural design phase is in charge of defining the most proper organizational structure for the MAS, i.e., the topology of interactions in the MAS and the control regime for the MAS which most effectively enables to fulfill the MAS goals. This definition of the organizational structure has to account for a variety of factors; including the need to somehow reflect the structure of the real-world organization in the MAS structure, the characteristics of the environment and of the patterns of access to it, the need of simplifying the enactment of the organizational rules, the need to fulfill any identified non-functional requirement, as well as the obvious need to keep the design as simple as possible. Once the most appropriate organizational structure is defined, the roles and interactions models identified in the analysis phase (which were preliminary, in that they were not situated in any actual organizational structure) can be finalized to account for all newly identified interactions and possibly for newly identified roles.

Past the architectural design phase, the detailed design involves identifying: (i) an agent model, i.e. the set of agent classes in the MAS, implementing the identified roles and the specific instances of these classes; and (ii) a services model, expressing services and interaction protocols to be provided within agent classes. The result of the design phase is assumed to be something that could be implemented in a technology-neutral way.

2.2. AUML IN A NUTSHELL

Several approaches address the problem of extending UML notation for agent-based systems. Some of these are: Agent UML [Odell et al., 2000; Bauer et al. 2000]; MESSAGE/UML (Methodology for Engineering Systems of Software AGEnts) [Caire et al., 2001]; AOR (Agent-Object Relationships) [Wagner, 2002]; PASSI [Cossentino and Sabatucci, 2004].

Among them, Agent UML (AUML) is particularly interesting because it builds on the acknowledged success of UML in supporting industrial-strength software engineering.

The core part of AUML is the Agents Interaction Protocol (AIP), specified by means of protocol diagrams. Protocol diagrams extensions to UML include agent roles, multithreaded lifelines, extended message semantics, parameterized

nested protocols, and protocol templates. AUML also proposes other extensions to UML in order to better capture more complete role specifications, packages with agent interfaces, deployment diagrams indicating mobility, emergence, etc. However, those notations are less rich than the ones proposed for AIP.
A three-layer representation of the AIP is defined as patterns representing both the message communication between agents and the corresponding constraints on the content of such messages.

1. The first layer represents the communication protocols. Protocols are modeled as whole entities (expressed by means of an enriched sequence diagram) that facilitates protocol reuse. Each protocol is treated as a package which aggregates modeling elements into a conceptual whole. Moreover, packages may be nested, so it is possible to define more reusable and easy to assemble protocols. AUML considers an AIP as a template, a parameterized model element whose parameters are defined at model time. These parameters are divided in three categories: roles, constraints, and communication acts.
2. The second layer represents interactions. The inter-agent interactions implementing the protocol are described using sequence diagrams, activity diagrams and statecharts that include some extensions to UML. Thus, these make possible to represent Agents (eventually specifying their Class) and their Roles. Also, instead of traditional OO messages the enhanced sequence diagrams specify the agent communication act. Moreover, an extension supports concurrent threads of interactions (considering: concurrence, inclusive or, and exclusive or). The activity diagram conveys operations and events that specify an explicit thread of control into an AIP. This process-centric view is particularly useful for complex interaction protocols that involve concurrent processing. Finally, the statecharts are useful for expressing constraints for AIP.
3. In the third layer activity diagrams and statecharts are used to specify the internal behavior of an Agent.

2.3. MAS ABSTRACTIONS AND GAIA + AUML

In other studies [Cernuzzi et al., 2004; Cernuzzi and Zambonelli, 2004] we have proposed to integrate the standard AUML notation into the Gaia process in order to make Gaia more expressive and easier to be accepted by software engineers. The proposal is mainly centered on adopting AIP notations from AUML instead of the Gaia ones. For simplicity and space reasons we call them Gaia+AUML. Thus, in this section we analyze the adequacy of Gaia+AUML for specifying all the proposed abstractions.

- *Roles*

Roles have been considered as the basis for agent design, since agents act by playing specific roles. AUML proposes some graphical notations, not always formal, for role modeling, as follow:

i) a general class diagram which may be used to identify the tasks a role have to accomplish with. An agent may play different roles in the same sequence diagram and each node is a class of role;
ii) collaboration diagrams, in which each communication act may be labeled with the related role;
iii) activity diagrams, which may represent agent roles by associating them to each activity; moreover, changes in roles may be represented on activity diagrams using notes.

Additionally, the role specification in Gaia is more expressive, more formal and includes more relevant aspects than the AUML proposal. Specifically, Gaia includes the description of permissions and responsibilities into role modeling. Permissions specify the resources that are available for playing that role and, doing so, they create direct relationships between the role activities and the environment. The responsibilities of a role, on the other hand, define the role's actual functionality. There are two types of responsibilities: safety properties and liveness properties. Safety properties are properties that the agent acting in the role must always preserve and are expressed as predicates over the variables/resources in the permissions of the role. Liveness properties describe generalized behavioral patterns of the role and are represented by a regular expression over the sets of activities and protocols the role executes. In this sense, role modeling is quite rich compared to other similar models in alternative AOSE methodologies and we may conclude that Gaia+AUML adequately covers this abstraction.

- *Activities and Goals*

The activities and goals that characterize roles are related to the role specification. While both try to capture functional requirements, goals have been considered to be more general and stable than activities to specify the general behavior of the system to be. Both Gaia and AUML cover activities specification, but Gaia considers them as a part of the role diagrams in which both activities (tasks that an agent playing that role can accomplish by itself) and protocols (tasks that involve coordination with other agents playing specific roles) are captured. In AUML activity diagrams and statecharts are used to specify the internal behavior of agents and its activities, nevertheless, neither Gaia nor AUML

are focused on goal specification. Only a limited specification of the general goals of each role is provided in Gaia by means of informal descriptions, and for this reason, we may conclude that Gaia+AUML may be improved to better cover this abstraction.

- *Agents*

Agent is the first class abstraction in the paradigm. As expected, both Gaia and AUML support the definition of agents using some kind of agent class diagram and allow the specification of which roles a specific agent may play during its lifetime. Therefore, we may conclude that Gaia+AUML adequately covers this abstraction.

- *Interaction with the Environment*

Normally, the context in which agents operate is composed not just of other agents, but also of human interactions, objects, and other kinds of resources. To capture those interactions, AUML offers the use of sequence diagrams that consider the interaction with the environment objects. However, this may cause the designers to lose abstraction consistency when modeling objects, humans or other entities in addition to agents in the same model and with the same stereotypes. Gaia covers the interaction with the environment mainly capturing the resources that the agents in the organization need for accomplishing their functions. However, the notation is not entirely formal nor standardized, and thus, it is clear that Gaia+AUML needs some improvement to better cover this abstraction.

- *Organizations and sub-organizations*

A lot of methodologies consider that an organization is just a set of roles; however, the same set of roles may assume very different types of organizations (i.e. hierarchy, network of peer, peers with a coordinator, etc.). Although Gaia explicitly considers structures of the overall organizations and sub-organizations, its notations are not well defined and do not offer specific guidelines for making design decisions. AUML [Parunak and Odell 2002] presents some problems with organizational structures notation. In fact the use of class diagrams for the overall organization, subclasses (composition relationship) representing the sub-organizations, and inheritance for specific agents (that tend to consider an organization as a simple collection of roles) may be considered quite poor notations. Moreover, according to Huget, Odell, and Bauer [UML and Agents: Current Trends and Future Directions], even if groups are already considered through UML based diagrams, they are still immature. Therefore, we may conclude that Gaia+AUML mainly covers this abstraction but needs more specific notations and guidelines.

- *Organizational rules*

The organizational structure may be insufficient to specify the general behavior of the MAS in terms of the interaction among the agents in the society, and for this reason, some methodologies have proposed to specifically capture the rules governing this organization. Gaia considers the organizational rules in a perspective that is coherent with the notion of responsibilities defined for roles, but referring to the organization as a whole. Accordingly, it is also possible to distinguish between safety and liveness organizational rules. Gaia adopts a formal defined notation based on regular expressions for rule specification, which is probably the more specific and formalized notation among the AOSE proposals. Therefore, we may conclude that Gaia+AUML adequately covers this abstraction.

- *Interaction between agents (Protocols)*

Normally, agents operate in a context in which they need to cooperate, compete or just communicate (interact) with other agents. For this purpose AIP notation, proposed by AUML and adopted in Gaia+AUML, is now a *de facto* standard in the agent community. As we introduced in section 2.2, AUML considers 3 levels to represent protocol diagrams (*communication protocol, interaction among agent, and internal agent processing*).
Therefore, we may conclude that Gaia+AUML efficiently covers this abstraction.

3. THE CENTRAL BANK AUDITING CASE STUDY

A multi-layer agent architecture for remote auditing in public institutions has been proposed to solve some deficiencies of the Central Bank of Paraguay (BCP – Banco Central del Paraguay), which is in charge of the auditing process of the financial-account management of credit firms. As the reader may observe in the proposed model of Figure 1, the BCP sends different agents to the financial institutions that the BCP needs to audit. In each financial institution the files are validated and modified to ensure correctness of information and format. Once the proper agent has completed its auditing activities, the obtained files are compressed and encrypted before being sent back to the BCP. Finally, the BCP sends an electronic receipt to the audited institution.

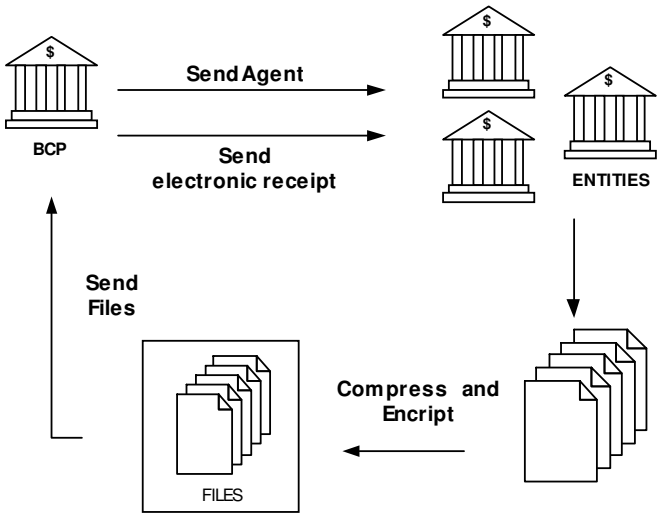


Figure 1. The proposed model

The experimental results achieved with the Auditing Agents BCP system, solve the problems identified in the current working model of the BCP.

3.1. THE BCP AUDITING SYSTEM IN GAIA + AUML

Possible sub-organizations
The auditing system includes four different processes: the correct format verification, the information validity verification, the verification of the relationships between files (also called referential integrity), and the verification of the sum of the debit for each client. Each verification process operates on different data files and includes multiple check activities, one for each of these four different files. Thus, it is natural to think in four different sub-organizations, one for each verification process.

Environmental Model
In this application the environmental model comprises the data bases of the audited banks (accessible if the agent has the corresponding permissions), a set of reports generated by the different agents, and the data base of the BCP.

Preliminary Roles Model
In this model designers capture the basic functional requirements of the application in terms of the roles that the different actors can play. Independently from the organizational structure, the main role is the Auditor which will be in charge of the general auditing process. Moreover, other roles are those related to each of the four verification processes previously identified. Other roles would may appear in the design phase. For space reasons we just present the Auditor role schema (see Figure 2).

Role Schema: Auditor

Description:
Move to the audited bank and interact with the checker in order to identify the agent. If validated, it starts the auditing process coordinating the verification agents (format, validity, relation, and sum). Once it receives all the correct reports, it encrypts the DB and goes back to the BCP.
If it is not validated, the auditing process is not allowed and the auditor has to go back to the BCP. Otherwise, each error detected during the auditing process is registered in a special file. The auditor stays in the bank until all the errors have been fixed.

Protocols y Activities:
ReceiveConfiguration, SendKey, ReceiveConfirmation, RequestAuditFormat, RequestAuditValidity, RequestAuditRelation, RequestAuditSum, ReceiveFormatReports, ReceiveValidityReports, ReceiveRelationReports, ReceiveSumReport, EncriptBD, GenerateFinalReport

Permisssions:
reads: configuration // user configuration
confirmation //boolean
FormatReports (CA, CB, CC, CD)
ValidityReports (CA, CB, CC, CD)
RealtionReports (CA, CB, CC, CD)
SumReport
AuditedBD (CA, CB, CC, CD)
generates: FinalReport //the auditing general report

Responsabilities:

Liveness:

- AUDITOR = ReceiveConfiguration.SendKey.receiveConfirmation. FORMAT.VALIDITY.RELATION.SUM.(Error || EncriptBD. GenerateFinalReport)
- FORMAT = RequestAuditFormat.ReceiveFormatReports
- VALIDITY = RequestAuditValidity.ReceiveValidityReports
- RELATION = RequestAuditRelation.ReceiveRelationReports
- SUM = RequestAuditSum.ReceiveSumReport

Safety:

- $confirmation = true \Rightarrow informeFinal \neq null$
- $ReportFCA=true \wedge ReportFCB=true \wedge ReportFCC=true \wedge ReportFCD=true \wedge ReportVCA=true \wedge ReportVCB=true \wedge ReportVCC=true \wedge ReportVCD=true \wedge ReportRCA\text{-}CC=true \wedge ReportRCB\text{-}CA=true \wedge ReportRCB\text{-}CC=true \wedge ReportRCC\text{-}CA=true \wedge ReportSum = true \Rightarrow encript(Audited\ BD)$
- $confirmation = false \Rightarrow error$

Figure 2. The auditor role schema

Preliminary Interaction Model
As for the preliminary roles model, and although this model may experience changes once the organizational structure is defined, some preliminary interaction protocols could be identified. For this model we adopt the AIP notation from AUML, and a representative example of this is presented in Figure 3 (for space reasons all the formats, validities and relations verifications are grouped in one agent class for each type).
Neither Gaia's nor AUML's notations offer facilities to specify agent mobility. Therefore, to represent agent migration we proposed the adoption of dashed arrows and of dashed boxes (e.g. in figure 3 the agent Auditor moves from the BCP to other bank).

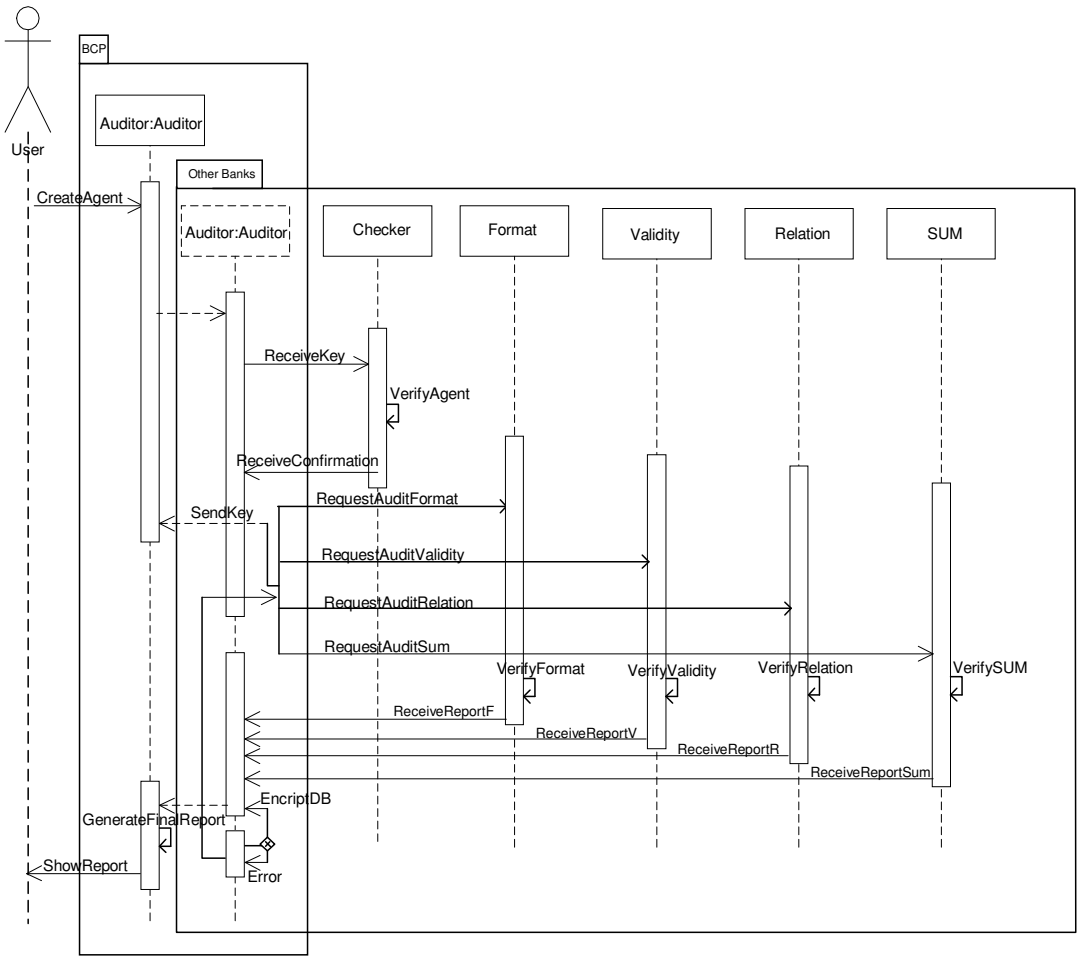


Figure 3. An AIP protocol model

Organizational Rules

The organizational rules are invariants that agents, playing the corresponding roles, have to observe for maintaining a correct behavior of the organization.

Different kind of rules may be defined for the BCP auditing system, but for space reasons we just present some examples of the complete set.

Some of these rules may also control the order of the activities. For example, rule 1 means that CreateAgent has to be executed before of ProgrammingAgent; while rule 2 states that ReceiveConfirmation precedes the warning for an ErrorKey or the request for a specific auditing process (in this case the format auditing).

Other rules may specify different constraints for the general behavior of the organization. For example, rule 4 means that if an Auditor has been validated all the corresponding reports (for Format, Validity, Relation and Sum) must exist.

1. CreateAgent $\rightarrow$ ProgramminAgent
2. ReceiveConfirmation $\rightarrow$ ErrorKey | RequestAuditFormat
3. Key $\neq$ null
4. $\forall$ Auditor, confirmation=true $\rightarrow$ ReportFCA $\neq$ null $\wedge$ ReportFCB $\neq$ null $\wedge$ ReportFCC $\neq$ null $\wedge$ ReportFCD $\neq$ null $\wedge$ ReportVCA $\neq$ null $\wedge$ ReportVCB $\neq$ null $\wedge$ InformeVCC $\neq$ null $\wedge$ ReportVCD $\neq$ null $\wedge$ ReportRCA-CC $\neq$ null $\wedge$ ReportRCB-CA $\neq$ null $\wedge$ ReportRCB-CC $\neq$ null $\wedge$ ReportRCC-CA $\neq$ null $\wedge$ ReportSum $\neq$ null
5. $\forall$ Auditor, home=true $\wedge$ FinalReport $\neq$ null $\rightarrow$ change BD BCP

Organizational Structure

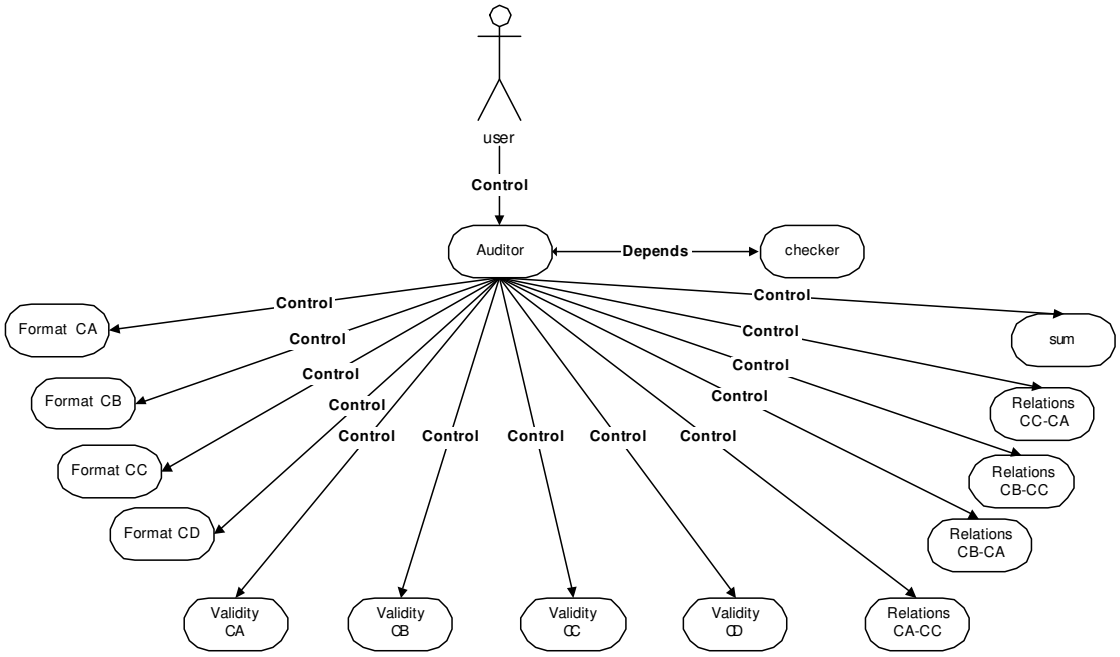


Figure 4. Organizational structure

Since the user has the control over the system, she/he is in charge of the creation of the auditor agent. This auditor agent in turn, depends on the checker agent that is in charge of auditor validation. The auditing process starts once the auditor creates all the agents in charge of the different verification processes for the audited bank. Those agents prepare and send a report according to the corresponding validation results. Since the auditor has created all the verification agents, it controls those agents and may eventually decide to eliminate them at any moment.

Detailed Design

The detailed design of Gaia considers the Agent Model, in which each role corresponds to a class, and the Service Model. For each auditing process, one Auditor and one Checker agents are instantiated while the other agents (Format, Validity, Relation, and Sum) may or may not be instantiated. If the Auditor agent is not recognized by the Checker these other agents are not instantiated, in all other cases, the verification agents are created when necessary until all errors have been corrected.

In the Service Model the designers identify the services associated to each class specifying inputs, outputs, pre and post-conditions. Inputs and outputs are derived from AIP (for the services associated to the agent interaction) and from the Environmental Model (for the services that operate on the resources). Pre and post-conditions derive from the safety properties of roles, from the organizational rules, from constraints on the availability of the resources, and from specific values of resources or data from other agents. For example the service "RequestAuditFormat" needs the "User Configuration" as input; it has the pre-condition "Confirmation=True", and the post-condition " FormatCA• NULL and FormatCB• NULL and FormatCC• NULL and FormatCD• NULL".

Gaia and AUML do not cover the development and implementation stages. For those activities we adopt the Aglets Software Development Kit (ASDK).

4. DISCUSSION

During these last two years, our group has designed and developed different multi-agent systems using Gaia and Gaia+AUML. The Auditing system for the BCP is a representative example of them. Based on this experience, it is possible to analyze the advantages and limitations of Gaia+AUML in the analysis and design of MAS modeling common abstractions.

As already discussed in section 2.3, Gaia offers a general and coherent framework for the design and specification of MAS while Gaia+AUML adequately covers the roles, agents, organizations and sub-organizations, organizational rules, and interactions between agent abstractions. Moreover, the Gaia+AUML models and notations for roles and organizational rules are richer than those proposed by others methodologies. More specifically, both roles and organizational rules models adopt the permission and responsibilities notions that may formally express relevant specifications of the system's properties in a descriptive way. Also, the adoption of AUML notations for the specification of the interactions between agents is a trend in AOSE methodologies that has demonstrated to be useful in guiding the transformation from the design to the development of MAS.

Some limitations of Gaia+AUML are related with the notations adopted, in particular, the specification of the interaction with the environment and of the organizations and sub-organizations need to be improved. However, in the AOSE arena there are no standards that properly cover these aspects. Furthermore, Gaia+AUML still needs specific notations to capture specific aspects. As an example of this, we have proposed an extension of AUML to specify agents mobility in the BCP auditing system, however, this proposal is far to be accepted as a standard in the AUML community.

Finally, Gaia+AUML also suffers from some limitations with regards to the goals specifications. As advocated by different authors, goals may specify the general behavior and consequently the functional requirements of the system-to-be in a more stable way. This flaw is related to the way the methodology covers the requirement elicitation phase. However, as presented in section 2.1, Gaia framework does not include requirement elicitation activities and we consider that improvements in this aspect are a strongly needed.

Taking these limitations into consideration, it may be interesting to briefly analyze the approaches adopted by other methodologies.

An approach quite common among the methodologies influenced by the object orientation paradigm is to adopt the Use Cases diagrams to specify requirements. Some examples of such methodologies are PASSI [Cossentino and Sabatucci, 2004], Roadmap [Juan et al., 2002], and MaSE [DeLoach et al., 2001]. Nevertheless, Use Cases diagrams are not enough to capture all the functional and non-functional requirements of a MAS.

Another approach focuses on the goals specification. Some examples of methodologies that follow this approach are Tropos [Bresciani et al., 2001; Giunchiglia et al., 2002] and Prometheus [Padgham and Winikoff, 2002]. Among all these, the most interesting proposal is Tropos in which the requirements phase is strongly influenced by the i* modeling framework [Yu, 1995].

Therefore, we may conclude that a possible way to improve Gaia's general framework is to consider the requirement elicitation phase; perhaps by adopting similar solutions to the Tropos one, or even better, by exploring alternative ways that can be integrated into Gaia while still maintaining its coherence and harmony.

5. CONCLUSIONS AND FUTURE WORKS

Authors have proposed different abstractions and methodologies to provide clear guidelines for the analysis, design, and development of MAS; offering specific notations for each model. Also, since agent-based computing includes a great variety of applications, it may be argued that different types of applications may imply different abstractions. However, independently of the application of any system in particular, our hypothesis is that some common abstractions may be identified in all applications using this paradigm. Consequently, this paper focuses on the common abstractions used to model the complexity of MAS: roles, activities with their corresponding goals, agents, interaction with the environment, organizational aspects (structure and control), and interaction between agents.

The present work also analyzes which of these abstractions are covered by Gaia integrated with AUML. Moreover, the paper presents, as a representative example, an agent based solution modeled in Gaia with AUML and developed for the auditing process of the Central Bank of Paraguay.

We may conclude that Gaia offers a general and coherent framework for the design and specification of MAS and that, despite some limitations related with the notations adopted and the lack of requirements specification, Gaia+AUML adequately covers the roles, agents, organizations and sub-organizations, organizational rules, and interactions between agents abstractions.

Therefore, possible lines that may be explored in future works are concerned with the integration into Gaia of an explicit requirements elicitation model and some improvements to Gaia and AUML notations.

Acknowledgements

Special thanks to Professor José Bogarin, who co-directed the BCP project with the first author, and Anouk Twijnstra and Cynthia Villalba for their collaboration in the development of the Auditing Agents System for the BCP.

References

Bauer, B., Müller, J., and Odell, J., 2000. Agent UML: A Formalism for Specifying Multiagent Software Systems. In: Ciancarini, P., and Wooldridge, M. (Eds.) Agent-Oriented Software Engineering - Proceedings of the First International Whorkshop (AOSE-2000). Springer-Verlag, Berlin (Germany) , pp. 91-103

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J., 2001. A Knowledge Level Software Engineering Methodology for Agent Oriented Programming. In: Proceedings of the 5th International Conference on Autonomous Agent*s*. ACM Press, Montreal (Canada), pp. 648-655

Caire, G., Chainho, P., Evans, R., Garijo, F., Gómez Sanz, J., Kearney, P., Leal, F., Massonet, P., Pavón, J., 2001. Agent Oriented Analysis Using MESSAGE/UML. Proceedings of Agent-Oriented Software Engineering – AOSE 01, May 2001, Montreal (Canada), pp. 101-107

Cernuzzi, L. and Rossi, G., 2002. On The Evaluation Of Agent Oriented Methodologies. Proceedings of the OOPSLA 02 - Workshop on Agent-Oriented Methodologies, November 2002, Seattle (USA), pp. 21-30

Cernuzzi, L., Juan, T., Sterling, L., and Zambonelli, F., 2004. The Gaia Methodology. (Chapter book) in *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering handbook.* F. Bergenti, M.-P. Gleizes and F. Zambonelli Editors, (pp. 69-88), Kluwer Publishing, 1-4020-8057-3

Cernuzzi, L., and Zambonelli, F., 2004. Experiencing AUML in the Gaia Methodology. In: Proceedings of 6th International Conference on Enterprise Information Systems - ICEIS 2004 (Vol. 3, pp. 283-288), Porto, Portugal, April 2004

Ciancarini, P. and Wooldridge, M., 2001. Agent-Oriented Software Engineering. Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering, Springer Verlag, LNCS, Vol. 1957, pp. 1-24

Cossentino, M. and Sabatucci, L., 2004. Agent System Implementation in Agent-Based Manufacturing and Control Systems: New Agile Manufacturing Solutions for Achieving Peak Performance. Paolucci M. and Sacile R. editors. CRC Press, April 2004

Cossentino, M. and Zambonelli, F., 2004. Multiagent Systems Development from the Autonomy Perspective, *Computational Autonomy*. LNCS Series No. 2969, Elsevier Ed.

DeLoach, S., Wood, M. and Sparkman, C., 2001. Multiagent Systems Engineering. *International Journal of Software Engineering and Knowledge Engineering*, vol. 11, No. 3, pp. 231-258

Giunchiglia, F., Mylopoulos, J. and Perini A., 2002. The Tropos Software Development Methodology: Processes, Models and Diagrams. Proceedings of Agent-Oriented Software Engineering (AOSE-2002), July 2002, Bologna (Italy), pp 63-74

Hoa Dam, K., and Winikoff, M., 2003. Comparing Agent-Oriented Methodologies. Proceedings of Agent Oriented Information Systems-AOIS'03, July 2003, Melbourne (Australia), pp. 78 - 93

Iglesias, C., Garijo, M. and González, J.C., 1999. A survey of Agent-Oriented Methodologies. In: Muller, J.P., Singh, M., and Roa, A.S. (Eds.), Intelligent Agent V, Proceeding of ATAL-98, Springer, LNCS 1555, pp. 317-330

Jennings, N., 2000. On agent-based Software Engineering, *Artificial Intelligence* 117, pp. 277-296.

Jennings, N. R., 2001, An Agent-Based Approach for Building Complex Software System, Communications of the ACM, Vol. 44, No. 4, pp. 35-41.

Juan, T., Pearce, A. and Sterling, L., 2002. ROADMAP: Extending the Gaia Methodology for Complex Open Systems. Proceeding of the First International Conference on Autonomous Agents and Multi-Agent Systems - AAMAS '02, July 15-19, 2002, Bologna (Italy), pp. 3-10

Odell, J., Parunak, H. v. D., and Bauer, B., 2000. Extending UML for Agents. Proceedings of Workshop on Agent Oriented Information Systems – AOIS 2000, Austin, USA

Padgham, L. and Winikoff, M., 2002. Prometheus: A Methodology for Developing Intelligent Agents. Proceedings of the First International Conference on Autonomous Agents and Multi-Agent Systems - AAMAS '02, Third International Workshop on Agent-Oriented Software Engineering AOSE-2002, July 15, 2002, Bologna (Italy), pp. 135-146

Sturm, A., Shehory, O., 2003. A Framework for Evaluating Agent-Oriented Methodologies. Proceedings of Workshop on Agent-Oriented Information Systems – AOIS 2003, 5th International Bi-Conference, July 14, Melbourne, (Australia), October 13, 2003 and Chicago, IL (USA), 2003, LNCS 3030, pp. 94-109

Wagner, G., 2003. The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior. *Information Systems*, Vol. 28, No. 5, July, 2003, Elsevier, pp. 475-504

Yu, E., 1995. Modelling Strategic Relationships for Process Reengineering. PhD thesis, University of Toronto, Department of Computer Science

Zambonelli, F., Wooldridge, M. and Jennings, N. R., 2003. Developing Multiagent Systems: The Gaia Methodology. *ACM Transaction on Software Engineering and Methodology*, vol. 12, No. 3, pp. 417-470

Certificación de la Forma Normal de la Reducción de Polinomios*

Gilberto Pérez, José M. Molinelli, y José L. Freire
Universidad de A Coruña, Dept. Computación, LFCIA,
A Coruña, España, 15071
{gilberto,molineli,freire}@dc.fi.udc.es

Abstract

This paper deals with the general problem of extracting programs from constructive proofs by using the computer-aided proof system **Coq**. For that, we have chosen a non trivial example: to obtain of the polynomial reduction and the normal form for multivariate polynomials.

First, we represent these polynomials as a setoid in **Coq**, then we implement the sum and the product proving that we are working with a commutative ring without zero divisors. After that, we choose an appropiate order on polynomials and define the concept of canonical polynomial in order to prove the well-founded propriety for that order. The third step consists in representing in **Coq** the polynomial reduction for a family of nonzero polynomials as well as its reflexive and transitive closure. We also include the proof that this polynomial reduction is a noetherian relation with respect to the underlying polynomial order. Once we have defined the normal form, we specify and give a constructive proof of the corresponding algorithm.

Finally, we use the **Coq** mechanism for automatic generation of certified programs, to obtain a functional program in *OCAML*.

Keywords: Proof-checker, Coq system, Program extraction, Formal methods, Computer Algebra.

Resumen

En este trabajo, se explora el mecanismo para la obtención de programas certificados a partir pruebas constructivas utilizando el sistema **Coq** [1]. Para ello, se estudia un caso no trivial: la obtención de la forma normal de la reducción de polinomios en varias variables.

En la primera etapa del trabajo se consigue representar en **Coq** el conjunto de los polinomios de varias variables como un *setoide*, y se implementan sobre él la suma y el producto, probando que dotan a este conjunto de una estructura de anillo conmutativo (no trivial) sin divisores de cero. En una segunda etapa, se define un orden adecuado para este conjunto de polinomios y se introduce el concepto de polinomio canónico para demostrar que este orden es bien fundado. A continuación, se especifica formalmente el concepto de reducción de polinomios mediante una familia de polinomios no nulos, así como su clausura reflexiva y transitiva, y se demuestra que esta reducción es una relación noetheriana respecto al orden de polinomios subyacente. Finalmente, se define la forma normal y se especifica y demuestra, de forma constructiva, el algoritmo para su obtención.

A partir de esta prueba, mediante el mecanismo de generación automática de programas de **Coq**, se obtiene un programa funcional certificado en *OCAML* [2].

Palabras claves: Verificador de pruebas, Sistema Coq, Extracción de programas, Métodos formales, Computación algebraica.

1 Introducción

El sistema de prueba **Coq** es una implementación del Cálculo de Construcciones Inductivas (CCI) realizada en el INRIA [1], que utiliza por defecto la lógica intuicionista. Es a la vez un lenguaje de programación y un sistema de prueba. Permite, por una parte, la especificación y síntesis de programas, y, por otra parte, la

*Financiado parcialmente por MCyT TIC2002-02859 y Xunta de Galicia PGIDT02TIC00101CT

representación y verificación de pruebas matemáticas. El sistema extrae automáticamente [3] el contenido constructivo de dichas pruebas y así se obtiene el programa funcional correspondiente al algoritmo probado.

Esta trabajo se inscribe dentro del marco de las interacciones entre los métodos formales y la computación algebraica. Podemos citar en esta dirección, por ejemplo [4, 5, 6].

No vamos a incidir aquí sobre la descripción y uso del sistema de prueba **Coq**. La formalización de polinomios se puede ver con más detalle en [4]. Para este desarrollo se ha utilizado la versión 8.0pl2 de Coq y el código completo de las pruebas está disponible en http://www.dc.fi.udc.es/staff/gilberto/

2 Polinomios en Varias Variables

2.1 Términos

Representaremos los términos como listas de naturales (los que corresponden a los exponentes de la variables).

```
Definition term := list nat.
```

Como trabajaremos normalmente con términos de n variables, implementamos un predicado ($full$) para su identificación.

```
Definition full (n : nat) (t : term) := length t = n.
```

Probamos en Coq que los términos forman el monoide $T^n := \{x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} | \alpha_i \in N, i = 1, \dots, n\}$ con el producto.

```
Fixpoint term_mult (t1 t2 : term) {struct t2} : term := match t1, t2 with
  | nil, x => x
  | x, nil => x
  | n1 :: t1', n2 :: t2' => n1 + n2 :: term_mult t1' t2' end.
```

Hay muchas maneras de ordenar los términos, entre los posibles órdenes, elegimos e implementamos inductivamente el orden lexicográfico sobre términos (Ttm).

```
Inductive Ttm : term -> term -> Prop :=
  | Ttm_car : forall (n1 n2 : nat) (u1 u2 : term),
      n1 < n2 -> length u1 = length u2 -> Ttm (n1 :: u1) (n2 :: u2)
  | Ttm_cdr :
      forall (n : nat) (u1 u2 : term), Ttm u1 u2 -> Ttm (n :: u1) (n :: u2).
```

Comprobamos que Ttm es un buen orden, es decir, que es un orden estricto, total y admisible. Utilizando la noción de accesibilidad [7] probamos que el orden Ttm es bien fundado. Asímismo formalizamos otras operaciones sobre términos, como la divisibilidad, el mínimo común múltiplo y el cociente, pues las necesitaremos más adelante.

2.2 Monomios

Axiomatizamos en Coq la estructura de cuerpo ($Q : Set$), de modo que permita utilizarcualquier especificación de cuerpo que eventualmente esté disponible en el sistema. En la extracción, para que el código sea ejecutable será preciso proporcionar una implementación concreta.

Formalizamos los monomios como pares formados por un coeficiente y un término, utilizando un tipo producto (*prod*).

```
Definition monom := (Q * term).
```

2.3 Polinomios

Implementamos el conjunto de los polinomios como un setoide, tipo base que parametriza todo nuestro desarrollo posterior. Para ello definimos los polinomios con respecto a un cuerpo base (Q), definiendo las operaciones y una relación de equivalencia que representa la igualdad matemática de polinomios. Respecto a la formalización de polinomios elegimos dos niveles de aproximación:

- En un primer momento, definimos polinomios como listas, no necesariamente ordenadas, de monomios, e implementamos sus operaciones algebraicas. Con esto conseguimos definiciones claras y sencillas, matemáticamente hablando, aunque ineficientes desde el punto de vista computacional.
- Más tarde, formalizamos los polinomios canónicos, así como nuevas versiones de las operaciones sobre estos polinomios, mostrando que son matemáticamente equivalentes a las anteriores. Por razones de eficiencia, las pruebas constructivas se hacen exclusivamente sobre polinomios canónicos.

2.3.1 Formalización

Dado un número natural n, definiremos polinomios en n variables $x_1, \ldots, x_n$ con coeficientes en el cuerpo **Q**. Representamos un polinomio a partir de una lista de monomios.

```
Inductive pol : Set :=
  | vpol : pol
  | cpol : monom -> pol -> pol.
```

Un polinomio cuyos monomios tienen todos n variables, se identifica mediante el predicado $full\_term\_pol$.

```
Inductive full_term_pol : pol -> Prop :=
  | full_term_pol_nil : full_term_pol vpol
  | full_term_pol_cons : forall (t : term) (a : Q) (p : pol),
      full n t -> full_term_pol p -> full_term_pol (cpol (a, t) p).
```

Este tipo de dato para representar polinomios permite definir de modo simple sus operaciones. Por ejemplo, la suma de polinomios ($+_p$) consiste en la concatenación de polinomios.

```
Fixpoint suma_app (p q : pol) {struct p} : pol := match p with
  | vpol => q
  | cpol m1 p1 => cpol m1 (suma_app p1 q) end.
```

La relación de igualdad de polinomios ($=_p$) (**igualdad extensional**), compatible con ($+_p$), se formaliza en **Coq** mediante los constructores de la definición inductiva del predicado *equpol*.

```
Inductive equpol : pol -> pol -> Prop :=
  | equpol_refl : forall p : pol, equpol p p
  | equpol_sym : forall p q : pol, equpol p q -> equpol q p
  | equpol_tran : forall p q r : pol, equpol p q -> equpol q r -> equpol p r
  | equpol_cm : forall (m : monom) (p q : pol),
      equpol (cpol m (suma_app p q)) (suma_app p (cpol m q))
  | equpol_ss : forall p p' q q' : pol,
      equpol p p' -> equpol q q' -> equpol (suma_app p q) (suma_app p' q')
  | equpol_smt : forall (k k' : Q) (t : term) (p : pol),
      equpol (cpol (k, t) (cpol (k', t) p)) (cpol (plusQ k k', t) p)
  | equpol_elim :
      forall (m : monom) (p : pol), z_monom m -> equpol (cpol m p) p.
```

El producto de polinomios se define utilizando el producto de un monomio por un polinomio.

```
Fixpoint mult_p (p q : pol) {struct p} : pol := match p with
  | vpol => vpol
  | cpol m1 p1 => suma_app (mult_m q m1) (mult_p p1 q) end.
```

Asímismo se define en Coq el polinomio opuesto y se prueba que $(Q[x], +, .)$ es un anillo conmutativo que no tiene divisores de cero.

2.4 Polinomios en Forma Canónica

Hasta aquí hemos trabajado con polinomios no necesariamente ordenados, con coeficientes no necesariamente distintos de cero y con términos eventualmante repetidos. Ahora definimos, mediante un predicado, los polinomios canónicos como "polinomios ordenados, con coeficientes no nulos y sin términos duplicados".

En primer lugar, para ordenar los monomios necesitamos un predicado, low_pol, que determina si un término es mayor que el del monomio de cabeza de un polinomio dado. Dado un término t y un polinomio p, denotamos este predicado como ($low\_pol\ t\ p$).

```
Inductive low_pol : term -> pol -> Prop :=
  | low_nil : forall t : term, low_pol t vpol
  | low_cons : forall (t u : term) (x : Q) (p : pol), Ttm u t -> low_pol t (cpol (x, u) p).
```

Dado que hemos probado que el orden sobre términos ($<_L$) es total y bien fundado, es posible representar los polinomios como listas ordenadas, de monomios no nulos en forma decreciente. El predicado $full\_pol$ "identifica" los polinomios que están en forma canónica.

```
Inductive full_pol : pol -> Prop :=
  | full_pol_nil : full_pol vpol
  | full_pol_cons : forall (t : term) (a : Q) (P : pol), ~ eqQ a 0Q ->
      full n t -> full_pol P -> low_pol t P -> full_pol (cpol (a, t) P).
```

Probamos en Coq que todo polinomio admite, para un orden dado entre los términos, una única representación canónica. Llamamos *fun_can* a la construcción de la forma canónica de un polinomio en n variables que está incluida en la prueba del resultado anterior. Comprobamos que, efectivamente, el polinomio obtenido al aplicar la función fun_can a un polinomio, es un polinomio en forma canónica igual al dado.

```
Lemma can_fun : (p:pol)(full_term_pol p)->{q:pol|(full_pol q)/\(equpol p q)}.
Definition fun_can : pol -> pol.
```

$$(full\_pol\ (fun\_can\ p)) \ \wedge \ (p\ =_p\ (fun\_can\ p))$$

2.5 Elementos Destacados

En primer lugar, definimos la función que extrae y devuelve el coeficiente de un término dado en un polinomio.

```
Fixpoint coef (p : pol) (t : term) {struct p} : Q :=
  match p with
  | vpol => 0Q
  | cpol (c, u) p1 => match eq_tm_dec t u with
      | left x => plusQ c (coef p1 t)
      | right y => coef p1 t
      end
  end.
```

Decimos que un término $t \in T$ figura en un polinomio $p \in Q[X]$ si, y solo si, su coeficiente en ese polinomio no es igual a cero.

```
Definition term_in_pol (p : pol) (t : term) : Prop := ~ eqQ (coef p t) 0Q.
Definition no_term_in_pol (p : pol) (t : term) : Prop := eqQ (coef p t) 0Q.
```

Utilizando la noción de representante canónico de un polinomio p; se formaliza la definición de monomio principal de la forma siguiente: si es el polinomio nulo ($vpol$) a p se le asigna $(0, x_1^0 \ldots x_n^0)$, y en otro caso le asignamos el primer monomio de su representante canónico. A partir de esta, se definen los conceptos de término y coeficiente principal.

```
Definition hmonom (p : pol) : monom := match fun_can p with
  | vpol => (0Q, n_term_0 n)
  | cpol m q => m end.
Definition hterm (p : pol) := mon_term (hmonom p).
Definition hcoef (p : pol) := mon_coef (hmonom p).
```

Probamos resultado técnicos sobre estos elementos necesarios para la implementació de la reducción de polinomios.

3 Orden sobre Polinomios

Precisamos implementar una "reducción" sobre polinomios, la cual puede ser interpretada como un tipo de "congruencia dirigida", reemplazar polinomios por otros "más pequeños". Por ello se implementa, en primer lugar, un orden estricto sobre polinomios cualesquiera, basado únicamente en el orden admisible de los términos del polinomio y se verifican a continuación sus principales propiedades. Para demostrar que este orden es **bien fundado** se formaliza el "tipo polinomio canónico" y sobre este tipo se restringe el orden anterior.

3.1 Definición y Propiedades

Cualquier orden admisible sobre términos se puede extender a un orden estricto parcial sobre polinomios. En nuestro caso extendemos el orden lexicográfico sobre términos, que hemos probado que es admisible, de la manera siguiente.

Sea $<_L$ el orden lexicográfico sobre términos (T^n), y dados dos polinomios $p, q \in Q[x_1, \ldots, x_n]$,

$$p \prec q :\Longleftrightarrow \begin{cases} hterm\ (p)\ <_L\ hterm\ (q) \\ o \\ \exists t \in T^n;\ [H(p,t) = H(q,t)\ \wedge\ t \notin p\ \wedge\ t \in q] \end{cases}$$

donde:

$$H(p,t) := \sum_{(u \in p)\ \wedge\ (t <_L u)} (coef\ u\ p).u$$

La traducción en Coq, se hace mediante la formalización del siguiente predicado inductivo:

```
Inductive Tpol_Lex3 : pol -> pol -> Prop :=
  | Tpol_Lex3_v : forall (m : monom) (p : pol),
      full_mon n m -> full_term_pol p -> Tpol_Lex3 vpol (cpol m p)
  | Tpol_Lex3_car : forall (m1 m2 : monom) (p1 p2 : pol),
      full_mon n m1 ->full_mon n m2 ->full_term_pol p1 ->full_term_pol p2 ->
      Ttm (mon_term m1) (mon_term m2) -> Tpol_Lex3 (cpol m1 p1) (cpol m2 p2)
  | Tpol_Lex3_cdr : forall (k1 k2 : Q) (t : term) (p1 p2 : pol),
      full n t ->full_term_pol p1 -> full_term_pol p2 ->
      Tpol_Lex3 p1 p2 -> Tpol_Lex3 (cpol (k1, t) p1) (cpol (k2, t) p2).
```

Comprobamos que *la relación que acabamos de definir, es estable por la relación equpol sobre polinomios canónicos*, que son los que se utilizarán en el cálculo de la forma normal de la reducción. También verificamos que la relación anterior es un orden estricto y nos interesamos particularmente por una serie de propiedades de la relación $Tpol\_Lex3$ concernientes a polinomios canónicos, que son indisociables de la definición de $\prec$.

En general la extensión de $<_L$ a $\prec$ no es un orden total. Sin embargo probaremos que es bien fundado.

3.2 Tipo Polinomio Canónico

Anteriormente hemos definido el predicado $full\_pol$ para caracterizar los polinomios ordenados y sin coeficientes nulos. A partir de dicho predicado, damos una formalización del tipo polinomio canónico. Este tipo será particularmente útil para definir un orden bien fundado sobre polinomios y por tanto el tipo sobre el que se sustentará la formalización del cálculo de la forma normal de la reducción.

Un polinomio (pol) se dice canónico si verifica el predicado $full\_pol$, es decir está ordenado en orden decreciente de términos y con todos los coefientes no nulos.

```
Definition pol_full := {p : pol | full_pol p}.
```

Se utilizará $Q_c[X]$ para denotar el conjunto de los polinomios canónicos.

Ahora restringimos el orden $Tpol\_Lex3$ formalizado sobre el tipo pol, al nuevo tipo pol_full. Esta relación se denotara por $\prec_c$.

```
Definition Tpol_full (a b : pol_full) : Prop :=
  let (p, H1) return Prop := a in
  let (q, H2) return Prop := b in Tpol_Lex3 p q.
```

Definimos la función que devuelve la lista de términos de un polinomio cualquiera y así demostramos que el orden dado sobre polinomios canónicos, $\prec_c$, se conserva al pasar al orden lexicográfico de sus listas de términos. Utilizando que el orden lexicográfico de listas es bien fundado se prueba que el orden definido sobre polinomios canónicos también lo es.

```
Theorem Tpol_full_wf : well_founded Tpol_full.
```

4 Reducción de Polinomios

Generalizamos el algoritmo de la división de polinomios en una variable a polinomios en varias (n) variables, también llamado proceso de reducción. La principal diferencia es que se utiliza un algoritmo que divide un polinomio por un conjunto de polinomios.

4.1 Definiciones

Definición 4.1 *Dados $f, g, h \in Q[x_1, \ldots, x_n]$ con $f \neq_p vpol$, decimos que* **g se reduce a h módulo f** *en un sólo paso (denotado $g \xrightarrow{f} h$) si, y sólo si, existe un término $t \in g$ con $(hterm\ f)|t$, y*

$$h =_p g -_p f.{}_M \left(\frac{(coef\ t\ g)}{hcoef\ f}, \frac{t}{hterm\ f} \right)$$

```
Inductive red (f g h : pol) : Prop :=
    red1_simpl : forall t : term, term_in_pol g t -> full_term_pol g ->
      full_term_pol f -> full_term_pol h -> ~ equpol f vpol ->
      term_div (hterm f) t ->
      equpol h
        (suma_app g
           (pol_opp
              (mult_m f (divQ (coef g t) (hcoef f), div_term t (hterm f))))) ->
      red f g h.
```

Ejemplo 4.1 *Así por ejemplo, sean $<_L$ el orden lexicográfico definido por $(y <_L x)$, $g =_p 6x^3+4x^2y+4y^3-1$ y $f =_p 2xy + y^3$. Entonces,*

$$6x^3 + 4x^2y + 4y^3 - 1 \xrightarrow{2xy+y^3} h := g - f.\left(\frac{4x^2y}{2xy}\right) = 6x^3 - 2xy^3 + 4y^3 - 1$$

Aquí $4x^2y$ es el monomio de g que se cancela utilizando $(hterm\ f) = 2xy$.

En la definición anterior, el polinomio resultante h se puede pensar como el resto de la división de g por f en un paso, de manera similar a como se realiza en polinomios de una variable.

Otra caracterización de la reducción, muy útil para la simplificación de las pruebas cuando se tienen varias opciones (términos) de reducción, es formalizar la reducción explicitando el término que se simplifica.

Definición 4.2 *Dados $f, g, h \in Q[x_1, \ldots, x_n]$ y $t \in T^n$ con $f \neq_p vpol$, decimos que $g \xrightarrow{f;\, t} h$,* **g se reduce a h módulo f por el término t** *en un sólo paso, si, y sólo si, existe un término $u \in T^n$ con $t = u.(hterm\ f)$, $c =_Q \frac{(coef\ t\ g)}{hcoef\ f}$, y*

$$h =_p g -_p f.{}_M(c, u)$$

```
Inductive red_exp (f g h : pol) (t : term) : Prop :=
    red1_exp_simpl : term_in_pol g t -> full_term_pol g ->
      full_term_pol f -> full_term_pol h -> ~ equpol f vpol ->
      forall u : term, full n u -> t = term_mult u (hterm f) ->
      forall c : Q, eqQ c (divQ (coef g t) (hcoef f)) ->
      equpol h (suma_app g (pol_opp (mult_m f (c, u)))) -> red_exp f g h t.
```

La idea básica del algoritmo de la reducción en polinomios de varias variables es la misma que en la división de polinomios de una variable: cuando hacemos la reducción $g \xrightarrow{f;\, t} h$, queremos cancelar términos de g utilizando el término principal de f.

Con esta formalización probamos diversos resultados sobre la reducción como que

Lema 4.1 *Dados $f, g, f_i \in Q[x_1, \ldots, x_n]$ y $t \in T^n$, que verifican $f \xrightarrow{f_i;\, t} g$, entonces*

$$\exists h \in Q[x_1, \ldots, x_n] \text{ tal que } [(\forall t' \in h) \Rightarrow t <_L t'] \wedge (f -_p h) \xrightarrow{f_i} (g -_p h)$$

```
Lemma red_comp : forall (f g fi : pol) (t : term), red_exp fi f g t ->
 exists x : _, full_term_pol x /\
   (forall t' : term, term_in_pol x t' -> Ttm t t') /\
   red fi (suma_app f (pol_opp x)) (suma_app g (pol_opp x)).
```

Para definir la reducción por un conjunto de polinomios se implementa, en primer lugar, el predicado inductivo *for_all_list_pol* que determina la lista mínima de polinomios que verifican una propiedad P.

```
Inductive for_all_list_pol (P : pol -> Prop) : list pol -> Prop :=
  | For_all_nil : for_all_list_pol P nil
  | For_all_cs : forall (F : list pol) (p : pol),
      P p -> for_all_list_pol P F -> for_all_list_pol P (p :: F).
```

Este predicado nos permite definir un conjunto finito de polinomios de n variables no nulos, definiremos dicho conjunto por su función característica utilizando el predicado $for\_all\_list\_pol$.

```
Definition full_fam := for_all_list_pol (fun p : pol => full_term_pol p /\ ~ equpol p vpol).
```

Con la ayuda de este predicado, podemos extender el proceso de reducción, a un conjunto de polinomios no nulos de n variables.

Definición 4.3 *Sean $g, h \in Q[x_1, \ldots, x_n]$, y F un conjunto de polinomios no nulos de n variables. Decimos que* **g se reduce a h módulo F**, *denotado*

$$g \xrightarrow{F} h$$

si, y sólo sí, existe $f \in F$ tal que $g \xrightarrow{f} h$

```
Inductive Red1 (F : list pol) (g h : pol) : Prop :=
    Red1_simp : full_fam F ->
      forall f : pol, pol_In_ensemb f F -> red f g h -> Red1 F g h.
```

Implementamos el proceso de reducción como la clausura reflexiva transitiva de la reducción por un conjunto de polinomios, denotado por $\xrightarrow[*]{F}$.

```
Inductive Red3 (F : list pol) : pol -> pol -> Prop :=
  | Red3_eq : forall g h : pol, full_fam F ->
      full_term_pol g -> full_term_pol h -> equpol g h -> Red3 F g h
  | Red3_step : forall g h : pol, full_fam F ->
      full_term_pol g -> full_term_pol h -> Red1 F g h -> Red3 F g h
  | Red3_trans : forall g f h : pol,
      full_fam F -> full_term_pol f -> full_term_pol g ->
      full_term_pol h -> Red3 F g f -> Red3 F f h -> Red3 F g h.
```

Se prueban diversos resultados técnicos relativos a la reducción antes definida. Por ejemplo, que $\xrightarrow[*]{F}$ es compatible con la multiplicación por un monomio.

```
Lemma Red3_mult_m : forall (f g : pol) (F : list pol), Red3 F f g ->
 forall (c : Q) (t : term), full n t ->
 Red3 F (mult_m f (c, t)) (mult_m g (c, t)).
```

Una condición suficiente para la reducción en una sola etapa es la siguiente.

Teorema 4.1 *Dados $f, g \in Q[x_1, \ldots, x_n]$, y F un conjunto de polinomios no nulos de n variables, se verifica*

$$\exists m \in M_{Q,n}, no\ nulo; \exists f_i \in F; (f =_p g - f_{i \cdot_M} m) \wedge [(hterm(f_{i \cdot_M} m)) \notin f] \Rightarrow\ g \xrightarrow{f} f$$

```
Lemma sufc_Red1 : forall (f g fi : pol) (F : list pol),
 full_fam F -> full_term_pol g -> full_term_pol f ->
 pol_In_ensemb fi F -> forall m : monom, full_mon n m -> ~ z_monom m ->
 equpol f (suma_app g (pol_opp (mult_m fi m))) ->
 no_term_in_pol f (hterm (mult_m fi m)) -> Red1 F g f.
```

4.2 Noetherianidad de la Reducción

La noetherianidad [8] es sin duda, entre las nociones clásicas del álgebra, la que tiene un tratamiento constructivo más delicado.

Examinando las exigencias de las matemáticas constructivas sobre la noetherianidad se aprecia que no se trata simplemente de proporcionar algoritmos pues hace falta también que las pruebas de su terminación sean constructivas. Se trata de una exigencia que roza la epistemología, y que es difícil de definir si no es por la práctica.

En nuestro caso, si queremos obtener el algoritmo que calcule la forma normal de un polinomio módulo un conjunto de polinomios, se necesita la propiedad de noetherianidad[1] de la reducción para garantizar la terminación de los cálculos.

[1]La noetherianidad garantiza la terminación de los cálculos sea cual sea la estrategia de reducción adoptada.

Definición 4.4 *Una relación R definida en un conjunto A es* **noetheriana** *si no existe una sucesión infinita de la forma $a_0\ R\ a_1\ R\ a_2\ R\ \ldots$.*

Otra definición posible para la noetherianidad es: **una relación R es noetheriana si, y sólo si, R^{-1} es bien fundada, siendo R^{-1} la relación simétrica de R** *($a\ R^{-1} b \Leftrightarrow b\ R\ a$).*

Pensamos que esta última definición de la noetherianidad es la más apropiada en nuestro caso y la formalizamos en **Coq** por:

```
Definition simetrico (A : Set) (R : A -> A -> Prop) (a b : A) : Prop := R b a.
Definition noetheriano (A : Set) (R : A -> A -> Prop) := well_founded (simetrico A R).
```

Para demostrar que la relación de reducción por un conjunto de polinomios dado es noetheriana, se necesita que el orden definido sobre polinomios sea bien fundado. Como esta propiedad sobre polinomios se ha probado sobre el tipo pol_full, necesitamos formalizar la notación de la reducción por un conjunto de polinomios para dicho tipo.

```
Definition inc_Red1 (F : list pol_full) (p q : pol_full) : Prop := Red1 (inc_list F) (inc p) (inc q).
```

El algoritmo de reducción de un polinomio f módulo un conjunto de polinomios F es no determinista. Sin embargo, siempre termina, es decir $\xrightarrow{F}$ es noetheriana. Esto se debe esencialmente al hecho que cuando en el paso de reducción se elimina un término t de g, entonces los términos t' de g que verifican $t <_L t'$, no cambian.

Teorema 4.2 *Sean $f, g \in Q_c[X]$ y F un conjunto de polinomios canónicos. Se verifica que*

$$f \xrightarrow{F} g \ \Rightarrow\ g \prec_c f$$

```
Lemma Red1_menor2 : forall (F : list pol_full) (f g : pol_full), inc_Red1 F f g -> Tpol_full g f.
```

Como consecuencia del lema anterior y del hecho de que $\prec_c$ sobre $Q_c[X]$ está bien fundada, obtenemos que la reducción por un conjunto de polinomios es una relación noetheriana, lo que garantiza la terminación de los cálculos sea cual sea la estrategia de reducción adoptada.

```
Lemma buen_ord_red1 : forall F : list pol_full, noetheriano _ (inc_Red1 F).
red in |- *; intros.
apply wf_incl with Tpol_full.
red in |- *; intros.
red in H.
apply Red1_menor2 with F; auto.
apply Tpol_full_wf.
Qed.
```

5 Forma Normal

Formalizamos el algoritmo para el cálculo de la forma normal de un polinomio módulo un conjunto de polinomios F. La terminación de este algoritmo está garantizada por la noetherianidad de $\xrightarrow{F}$, probada en la sección anterior. Es importante resaltar que esta forma normal no es única en general.

5.1 Definiciones y Propiedades Elementales

Definición 5.1 *Un polinomio $f \in Q[x_1, \ldots, x_n]$ está en* **forma normal** *(o forma reducida) módulo F si f no es reducible módulo F, es decir no existe un polinomio $g \in Q[x_1, \ldots, x_n]$ tal que $f \xrightarrow{F} g$. Se denotará por f_{-F}.*

```
Definition normal (F : list pol) (f : pol) : Prop := forall g : pol, ~ Red1 F f g.
```

Comprobamos la *extensionalidad de la relación normal con respecto a equpol*, utilizando la extensionalidad de $\xrightarrow{F}$.

```
Lemma normal_ext : forall (p q : pol) (F : list pol),
  normal F p -> full_term_pol p -> equpol p q -> normal F q.
```

Un resultado técnico esperado y que induce la notación $f \xrightarrow[*]{F} f$, utilizada por algunos autores [9], para expresar que el polinomio f está en forma normal, es el siguiente.

Teorema 5.1 *Dados $f, g \in Q[x_1, \ldots, x_n]$, y F un conjunto de polinomios no nulos de n variables, se verifica que*

$$(f \xrightarrow[*]{F} g) \wedge f_{-F} \Rightarrow (f =_p g) \wedge g_{-F}$$

```
Lemma normal_eq_aux : forall (F : list pol) (f g : pol), Red3 F f g ->
  normal F f -> equpol f g /\ normal F g.
```

Un resultado útil que se obtiene a partir de la implementación de *polinomio en forma normal* es que *el polinomio vpol está en forma normal con respecto a cualquier conjunto de polinomios*[2]. Se prueba por introducción de la negación.

```
Lemma vpol_normal : forall G : list pol, normal G vpol.
```

5.2 Procedimiento de Normalización

Demostramos resultados computacionales para obtener el algoritmo del cálculo de una forma normal de un polinomio módulo un conjunto de polinomios.

Necesitamos previamente probar un lema de decidibilidad de la reducción, que simplificará la prueba final.

Lema 5.1 *Dados $f \in Q_c[X]$, y F un conjunto de polinomios canónicos no nulos, se verifica*

$$\{\exists\, g \in Q_c[X];\ f \xrightarrow{F} g\} \vee \{f_{-F}\}$$

```
Lemma prev_calcula_fn3bis : forall (F : list pol_full) (f : pol_full), full_fam (inc_list F) ->
 {g : pol_full | Red1 (inc_list F) (inc f) (inc g)} + {normal (inc_list F) (inc f)}.
```

Teorema 5.2 *Dados $f \in Q_c[X]$ y F un conjunto de polinomios canónicos no nulos, se verifica*

$$\{\exists\, g \in Q_c[X];\ (f \xrightarrow[*]{F} g) \wedge g_{-F}\}$$

Prueba: Utilizando la recurrencia sobre la noetherianidad de la reducción por un conjunto de polinomios ($\xrightarrow{F}$) se obtiene la hipótesis de recurrencia noetheriana ($\forall\, y;\ f \xrightarrow{F} y \Rightarrow \{\exists\, y_0;\ (y \xrightarrow[*]{F} y_0) \wedge y_{0-F}\}$). Aplicando el lema anterior al polinomio f tenemos dos casos,

- $\exists\, p;\ f \xrightarrow{F} p$. Aplicando la hipótesis de recursión noetheriana al polinomio p, mediante la táctica **Elim**, se obtiene un polinomio en forma normal g, que verifica $p \xrightarrow[*]{F} g$. Así el polinomio buscado es g. Al obtener por la transitividad de *Red3* respecto al polinomio p, la meta $f \xrightarrow[*]{F} g$, deducimos que el polinomio buscado es g.
- f está en forma normal. El polinomio elegido es el propio f por la clausura reflexiva de $\xrightarrow{F}$.

```
Lemma calcula_fnbis : forall (F : list pol_full) (f : pol_full), full_fam (inc_list F) ->
 {g : pol_full | normal (inc_list F) (inc g) /\ Red3 (inc_list F) (inc f) (inc g)}.
intros.
cut (noetheriano _ (inc_Red1 F)); intros.
red in H0.
elim H0 with f; intros.
elim (prev_calcula_fn3bis F x); intros; trivial.
elim a; intros.
elim (H2 x0); intros.
split with x1.
elim p0.
split; trivial.
apply Red3_trans with (inc x0); auto.
```

[2]Algunos autores lo introducen como axioma.

```
inversion p.
inversion H4; auto.
split with x.
split; auto.
apply buen_ord_red1; auto.
Qed.
```

Llamamos for_norm a la construcción de la forma normal de un polinomio canónico módulo un conjunto de polinomios canónicos no nulos. La forma normal de un polinomio p, obtenida mediante esta función se denotará $(for\_norm\ F\ p)$.

```
Definition for_norm : list pol_full -> pol_full -> pol_full.
```

Probamos que, efectivamente, el polinomio obtenido al aplicar la función for_norm a un polinomio f canónico está en forma normal, verifica que $f \xrightarrow[*]{F} (for\_norm\ F\ f)$ y tiene el mismo número de variables que el inicial.

```
Lemma norm_corr : forall (F : list pol_full) (f : pol_full),
 full_fam (inc_list F) -> normal (inc_list F) (inc (for_norm F f)).
Lemma Red3_norm : forall (F : list pol_full) (f : pol_full),
 full_fam (inc_list F) -> Red3 (inc_list F) (inc f) (inc (for_norm F f)).
Lemma full_term_norm : forall (F : list pol_full) (f : pol_full),
 full_fam (inc_list F) -> full_term_pol (inc (for_norm F f)).
```

Probamos también diversas propiedades relativas a la estabilidad del ideal respecto a la forma normal.

Teorema 5.3 *Dados $g \in Q_c[X]$ y F un conjunto de polinomios canónicos no nulos, si la forma normal de g es vpol entonces g está en el ideal generado por F, es decir*

$$[(for\_norm\ F\ g) =_p vpol] \Rightarrow g \in < F >$$

```
Lemma pol_fn_id2 : forall (F : list pol_full) (g : pol_full), full_fam (inc_list F) ->
 equpol (inc (for_norm F g)) vpol -> Ideal (inc g) (inc_list F).
```

Teorema 5.4 *Dados $g \in Q_c[X]$ y F un conjunto de polinomios canónicos no nulos, entonces la forma normal de g, módulo F, está en el ideal generado por g y F, es decir*

$$(for\_norm\ F\ g) \in < g : F >$$

```
Lemma fn_p_id : forall (F : list pol_full) (g : pol_full),
 full_fam (inc_list F) -> Ideal (inc (for_norm F g)) (inc g :: inc_list F).
```

Teorema 5.5 *Dados $g \in Q_c[X]$ y F un conjunto de polinomios canónicos no nulos, entonces g está en el ideal generado por la forma normal de g y el conjunto de polinomios F, es decir*

$$g \in < (for\_norm\ F\ g) : F >$$

```
Lemma pol_fn_id : forall (F : list pol_full) (g : pol_full),
 full_fam (inc_list F) -> Ideal (inc g) (inc (for_norm F g) :: inc_list F).
```

Por último se prueba la decidibilidad de la comparación de la forma normal con el polinomio nulo ($vpol$)

```
Lemma for_norm_dec_vpol : forall (F : list pol_full) (g : pol_full), full_fam (inc_list F) ->
 {equpol (inc (for_norm F g)) vpol} + {~ equpol (inc (for_norm F g)) vpol}.
```

6 Extracción de Código

Una de las prestaciones más interesantes que ofrece Coq, y que además no está disponible en todos los sistemas, es la "extracción de código", es decir, la posibilidad de obtener automáticamente código ejecutable (código OCAML, en el caso de Coq) a partir de los resultados computacionales demostrados.

6.1 Implementación de la Parte Axiomatizada

Al haber axiomatizado las propiedades correspondientes al cuerpo base para el anillo de polinomios, la extracción de código no genera de modo automático una implementación de esta estructura. Por ello, para poder ejecutar los programas obtenidos, es preciso elegir un modelo de cuerpo (que cumpla los axiomas establecidos) y proporcionar una implementación adecuada para él en OCAML. En este trabajo hemos optado por utilizar directamente el tipo de dato *num* de la librería *Num* de OCAML, que constituye una implementación completa del cuerpo de los números racionales. Naturalmente, con esta elección hacemos descansar parte de la responsabilidad de la corrección del código en la de esta implementación. Pero esto, no supone en realidad un riesgo mucho mayor que el de confiar en la corrección del propio OCAML o del funcionamiento de la máquina en la que se ejecuta.

También es preciso fijar el número de variables con las que se va a trabajar (que es un parámetro del desarrollo realizado) y su nombre (para el "pretty-print"), lo que se consigue con la definición de dos valores: n y $variables$. (En el ejemplo que se muestra más abajo, se ha establecido $n = 6$ y $variables = [''x''; ''y''; ''z''; ''r''; ''s''; ''t''])$.

6.2 Optimización de la Entrada y Salida de Datos

Utilizar directamente la notación de OCAML para introducir y obtener valores del tipo de dato *pol*, que sirve para representar los polinomios, resultaría en extremo engorroso y podría llevar fácilmente a cometer errores en la introducción de datos y en la interpretación de los resultados. Por ello, se han añadido dos módulos al código generado automáticamente: uno que incorpora una función $string\_of\_pol : pol -> string$ que permite visualizar los valores de tipo *pol* de forma legible[3]; y otro que proporciona una función $poly : pol -> string$ que permite introducir estos valores como cadenas de caracteres, utilizando la notación que usa la función *string_of_pol* (sensiblemente flexibilizada para mayor comodidad). Así, por ejemplo, el polinomio $x^2y - \frac{2}{3}xy^2$ podría introducirse a partir de cualquiera de los siguientes *strings* (entre otros): $''1\ x2y\ -\ 2/3\ xy2''$, $''x2y\ -\ 2/3xy2''$, $''-2/3xy2\ +\ x2y''$, $''1/3xy2\ +\ x2y\ -\ xy2''$, $''xy\ -\ 2/3yxy''$. Con la misma intención, se han añadido algunas funciones más, como la función *list* que permite realizar la conversión entre las listas nativas de OCAML y el tipo correspondiente generado por Coq.

6.3 Ejemplo

```
# let normal l p = string_of_pol
                 (for_norm (list (List.map poly l)) (poly p));;
val normal: string list -> string -> string

# normal ["xy-y"; "-x+y2"] "xy2";;
-: string = "1 y4"

# let polset = ["126z5 +54z4 -49z3 +336z2 -576z";
  "1296yz -126z4 -486z3 + 49z2 - 168z";
    "9072y2 + 126z4 + 486z3 -49z2 + 168z -5184 ";
    "1944x -126z4 -486z3 + 49z2 -168z - 648"];;

# normal polset "57y - 59yx +45z2 - 8x2yz - 93x2z2 + 92z2y2";;
- : string =
"112/3 y + -22614121/1361367 z4 + 11231/1029 z3 + 366307003/3500658 z2 +
 -423909508/4084101 z"

# normal polset
"-15552x3yz - 15456y2z3 + 1008z5x2y - 36936y - 29160z2 + 1344x2yz2 + 87480xz2 -
9576yz - 59616z2y2 + 60264x2z2 + 4508z4y2 -4557z4x2 - 11592z6y2 + 28674yz3x +
7434z4yx + 178848xz2y2 + 9912xyz - 2891xz2y + 2793z2y + 5184x2yz - 27702yz3 -
```

[3]Las funciones de análisis léxico y sintáctico necesarias para este módulo han sido generadas con las herramientas camllex y camlyacc[10]

```
7560z3 +3888z4x2y + 149040yx - 392z3x2y - 114696yx2 - 180792x3z2 + 45198z5x2 -
7182z4y + 2205z4 - 21870z5 - 44712z5y2 - 5670z6 +15624x2z3 + 11718z6x2";;
- : string = "0"
```

7 Conclusiones

A través de este trabajo se puede confirmar que **Coq** es un sistema de ayuda a la prueba que obliga a hacer un esfuerzo hacia la estricta formalización y especificación de pruebas, lo cual permite transcribir esas pruebas más exactamente en lenguaje matemático. A la hora de hacer balance del uso de **Coq** en la teoría de polinomios, resaltamos los siguiente hechos.

- La necesidad de buscar una solución adecuada para el tratamiento de estructuras cocientes nos llevó a la conveniencia de trabajar con representantes canónicos que tuvimos que definir formalmente. Esta misma solución se está aplicando con éxito en otros proyectos de certificación.
- Los tipos inductivos de **Coq** permiten un grado de confortabilidad considerable, tanto para definir estructuras de datos como para definir predicados.
- Es necesario formalizar cuidadosamante los polinomios canónicos por razones de eficiencia.
- Son precisas varias caracterizaciones de la noción de reducción para su utilización en las distintas pruebas.
- En la prueba matemática empleamos muchos lemas intermedios sin reparar en ellos. El hecho de tener que probarlos en **Coq** profundiza nuestro conocimiento de las teorias que estamos trabajando.
- Se evidencia la necesidad de dar un orden bien fundado para garantizar la terminación del algoritmo de normalización.

Finalmente, nótese que la extracción produce programas legibles, reutilizables, eficaces y correctos.

References

[1] Dowek, G. et al. The Coq Proof Assistant Reference Manual. V8.0 INRIA 2004.

[2] Leroy, X. The Objetive Caml system release 3.08. Documentation and user's manual. INRIA 2004.

[3] Paulin-Mohring, C and Werner, B. Synthesis of ML programs in the system Coq. *J. of Symbolic Computation*. No. 15, pp 607-640. 1993.

[4] J.M. Barja y G. Pérez. Demostración en implementaciones concretas de anillos de polinomios. *Minisimposium Lógica, Matemática, Deducción Automática*. Actas RSME-2000, pág 7-25. Madrid, Enero 2000.

[5] Théry, L. A Certified Version of Buchberger's Algorithm. *Proceedings of Automated Deduction, CADE-15*,LNAI 1421, pp. 349-364. 1998.

[6] Medina, I. and Alonso, J.A. and Palomo F. Automatic verification of polynomial rings fundamental properties in ACL2. En *ACL2 Workshop*. 2000.

[7] Huet, G. Inductive Principles Formalized in the Calculus of Constructions. En *Programming of Future Generation Computers*, pp 205-216. Edit, K. Fuchi y M. Nivat. North-Holland, 1988.

[8] Paulson, L.C. Constructing Recursion Operators in Intuitionistic Type Theory. *J. of Symbolic Computation*. Vol II, No. 4, pp 325-355. 1993.

[9] Mishra, B. *Algorithmic Algebra*. Texts and Monographs in Computer Science. Springer-Verlag, 1993.

[10] Levine, J. and Mason, T. and Brown, D. *Lex and Yacc*. O'Reilly, Second edition, 1992.

Sistemas de Especificación de Transiciones Probabilísticas: Formato de Reglas y Bisimulación como Congruencia.

Ariel M. Fiuri
Universidad Nacional de Córdoba, Fa.M.A.F.,
afiori@hal.famaf.unc.edu.ar,
Motorola, GSG-Argentina,
ariel.fiuri@motorola.com
Córdoba, Argentina.

Abstract

In this paper I discuss the probabilistic transition systems, especially when its states are terms generated by some signature. The problem is to study when the equivalence relation of strong bisimulation is a congruence with respect to the functional symbols of that signature. I tackle this problem in three steps. First, I define a syntactic structure in a Plotkin [Plot81] style named probabilistic transition specification systems, the semantics of which are the probabilistic transition systems. Second, I study formats of rules in a tyft/tyxt style (Groote & Vaandraguer [GV92]) and I propose five different formats. Third, I present and prove a teorem that ensures that under those formats the strong bisimulation is a congruence. The importance of this result is based on the fact that those who need to check that the strong bisimulation is a congruence do not need to make expensive and error prone calculus, instead they only need to do a syntactic check proving that all rules that specify their model satisfy some of the proposed formats.

Keywords: Transition systems, Bisimulation, Congruence, Rule, Proof, Probability.

Resumen

En este trabajo estoy interesado en los sistemas de transición probabilísticos, especialmente cuando sus estados son términos generados por alguna signatura. Un problema con interés propio es el de estudiar cuándo la relación de equivalencia de bisimulación fuerte es una congruencia con respecto a los símbolos de función de la signatura. Ataco este problema en tres pasos. En primer término, defino una estructura sintáctica a lo Plotkin [Plot81] llamada Sistema de Especificación de Transiciones Probabilísticas, cuya semántica son los Sistemas de Transiciones Probabilísticos. En segundo lugar, estudio formatos de reglas al estilo tyft/tyxt de Groote & Vaandrager [GV92] y propongo cinco formatos diferentes. Por último, enuncio y pruebo un teorema que asegura que bajo estos formatos la bisimulación fuerte es una congruencia. Con este resultado, quien necesite chequear que la bisimulación fuerte es una congruencia no tendrá que hacer costosos cálculos propensos a errores, bastará con chequear sintácticamente que todas las reglas que especifican su modelo cumplen con algunos de los formatos propuestos.

Palabras claves: Sistema de transición, Bisimulación, Congruencia, Regla, Prueba, Probabilidad.

1. Introducción

En la vida real los sistemas a menudo ofrecen un proceder aleatorio o estocástico en su naturaleza. Por ejemplo, uno puede observar que un medio de comunicación distorsiona o pierde un mensaje el 2 % de las veces que transmite. Puede verse también que algún componente de una red se cae con una probabilidad $0,03$ o que en el proceso de fabricación de cierto producto, éste no pasa las pruebas de calidad con una probabilidad p.

Estas situaciones son descriptas por lenguajes probabilísticos. A un lenguaje probabilístico se le da semántica en términos de una clase especial de autómata llamado *Sistema Probabilísico Concurrente Rotulado*, y a su vez se le da semántica operacional con una estructura sintáctica llamada *Sistema de Especificación Probabilístico Concurrente Rotulado.*[1]

Cuando en el mundo real una falla o una posible mejora en la performance son detectadas, es posible querer cambiar un componente del sistema por otro "igual"pero más eficiente o con mejor comportamiento. En los Sistemas Probabilísticos Concurrentes Rotulados, una relación de equivalencia entre sus estados (llamada *bisimulación*) es la que fija el criterio de "igualdad".

Es importante poder asegurar que se puede cambiar un componente del sistema por otro y éste sigue comportándose como debiera. Esta es una propiedad del criterio de comparación llamada *congruencia.*

En este trabajo se encuentran condiciones en la semántica operacional que aseguran que el criterio de igualdad sea una congruencia. En la sección 2, se introducen los Sistemas Probabilísticos Concurrentes Rotulados (o sea la semántica del lenguaje probabilístico). En la sección 3, se definen los Sistemas de Especificación Probabilísticos Concurrentes Rotulados (o sea la semántica operacional del lenguaje). En la sección 4, se proponen condiciones para asegurar que la bisimulación sea una congruencia. En la 5, se demuestra que bajo estas condiciones la bisimulación es una congruencia y por último en la sección 6, se analizan brevemente algunos casos conocidos en la literatura de los sistemas concurrentes.

2. Sistemas Probabilísticos Concurrentes Rotulados ($SPCR$).

2.1. Definición

En esta sección se define el modelo semántico a especificar. Este introduce la noción de probabilidad sobre los sistemas de transición conocidos en la literatura. Definamos para ello la distribución de probabilidad sobre un conjunto dado, como es usual:

Definición 2.1 *Sea C un conjunto definimos $Distr(C)$ como el conjunto de distribuciones de probabilidad sobre C es decir $Distr(C) = \{f : C \to [0,1] | \sum_{e \in C} f(e) = 1\}$.*[2]

El modelo que manejaremos asigna a cada estado un conjunto de transiciones probabilísticas no determinísticas. Cada una de ellas representa un paso aleatorio del sistema. Estas tienen asociado un rótulo. Para un estado s, del conjunto de estados del sistema (S), manejaremos pares (a, μ) donde a es el rótulo y μ la distribución. En este tipo de sistemas la relación de transición $\to \subseteq S \times Act \times Distr(S)$, intuitivamente establece que para un estado s ejecutándose la acción a tendremos una probabilidad $\mu(t)$ de cambiar al estado t. Como está definido en [Baier], tenemos entonces:

Definición 2.2 *Un sistema de transiciones probabilístico concurrente rotulado con acciones (SPCR) es una terna $(S, Act, Steps)$ donde S es un conjunto de estados, Act un conjunto no vacío de rótulos y $Steps : S \to 2^{Act \times Distr(S)}$ una función que asigna a cada estado s un conjunto $Steps(s)$ de pares $(a, \mu) \in Act \times Distr(S)$ donde Distr(S) es el conjunto de funciones de distribución de probabilidades sobre los estados del sistema. Escribiremos $s \xrightarrow{a} \mu$ sii $s \in S$ y $a \in Act$ y $(a, \mu) \in Steps(s)$ y $s \xrightarrow{a} \mu$ será llamado una transición de s.*

Se representa gráficamente a los $SPCR$ como en la teoría clásica de autómatas, donde los estados son círculos y las transiciones son flechas desde un estado hacia un punto resaltado, el cual representa una distribución. Estas flechas están rotuladas con acciones. De este punto salen flechas hacia los estados que aparecen en la distribución, estando estas flechas rotuladas con las probabilidades de acceder a esos estados.[3]

2.2. Bisimulación

La relación de bisimulación es una relación semántica utilizada para comparar procesos. Es una relación de equivalencia que intuitivamente brinda la idea de que un proceso "proceda como" otro. Dos procesos

[1] La relación entre ambas semánticas es que una especifica a la otra, es decir que se le puede dar semántica a los Sistemas de Especificación Probabilísticos Concurrentes Rotulados en términos de los Sistemas Probabilísticos Concurrentes Rotulados

[2] Como notación usualmente se usará una modificación a la definición extensional de una función dada por:$\{(c_1, p_1), .., (c_n, p_n)\}$ denota la función $f : C \to [0,1]$ tal que $f(c_i) = p_i$ y $f(x) = 0$ para todo $x \neq c_i, 1 \leq i \leq n$.

[3] En el transcurso de este trabajo aparecen algunos ejemplos gráficos, por ej. fig. 1 pág. 3.

son bisimilares si uno simula al otro pero a su vez ese otro simula al primero. Más generalmente se usa para comparar sistemas de transición, pudiendo ver cuando un sistema es imitado por un segundo sistema y éste imitado por el primero.

Antes de avanzar en este concepto necesitamos la noción de distribución sobre un conjunto de estados:

Definición 2.3 *Sea* $\mu = \{(s_1, p_1), .., (s_n, p_n)\}$ *con* $\mu \in Distr(S)$ *una distribución entonces definimos la distribución aplicada a un conjunto como*

$$\mu[c] = \sum_{s \in c} p | (s, p) \in \mu$$

Tomemos la definición de bisimulación de Baier [Baier]. Intuitivamente si c es una clase de equivalencia de la bisimulación, un estado s es bisimilar a otro s' si cada vez que en s se ejecuta la acción a en s' también se puede ejecutar a y la probabilidad de que se cambie a estados en c es la misma en ambos casos, formalmente:

Definición 2.4 (Bisimulación) *Una bisimulación sobre un* $SPCR$ $(S, Act, Steps)$ *es una relación de equivalencia* $\mathcal{R}$ *sobre* S *tal que:*

$$\forall (s, s') \in \mathcal{R} : s \xrightarrow{a} \mu \Rightarrow \exists \mu' \in Distr(S) : (s' \xrightarrow{a} \mu' \ y \ (\forall c \in S/\mathcal{R} : \mu[c] = \mu'[c]))$$

Definamos además la bisimulación fuerte ($\underline{\leftrightarrow}$) *como la menor relación que incluye a toda otra relación de bisimulación, o lo que es equivalente como la unión de todas las relaciones de bisimulación.*[4]

Ejemplo 2.1 *En la figura 1 se muestran dos estados (r y s) de un* $SPCR$ *que son bisimilares. Vemos que* $s \xrightarrow{a} \eta$ *con* $\eta = \{(s_2, 2/3), (s_3, 1/3)\}$ *y* $r \xrightarrow{a} \mu$ *con* $\mu = \{(r_1, 1/3), (r_2, 1/3), (r_3, 1/3)\}$

Los estados encerrados por líneas de puntos pertenecen a una clase de equivalencia de $\mathcal{R}$ *(digamos* c_1*), mientras que los encerrados por líneas continuas a otra (digamos* c_2*).*

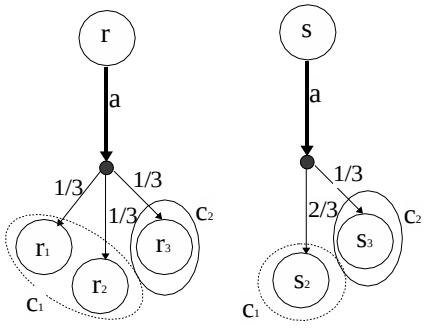


Figura 1: $SPCR$ donde s y r son bisimilares.

□

3. Sistemas de Especificación Probabilísticos Concurrentes Rotulados ($SEPCR$).

3.1. Definición

En esta sección, se define la estructura sintáctica que servirá para especificar a los $SPCR$. El interés está en aquellos $SPCR$ que sirven para dar semántica a algún lenguaje, es decir aquellos en los que los estados son términos de ese lenguaje. Al igual que en el trabajo de Groote y Vaandrager [GV92], se construye un sistema de pruebas cuyas reglas son también la semántica operacional del lenguaje. La novedad está en que este sistema extiende al de Groote y Vaandrager para poder representar transiciones probabilísticas.

[4] Notar que la bisimulación fuerte es también una relación de equivalencia. [LaSk89].

Tomemos tres conjuntos disjuntos de metavariables var, $pvar$ y μvar con elementos típicos $v, w, x, y, z...$, $\{p, q\}$ y $\{d, e\}$ respectivamente. Estos denotan términos del lenguaje subyacente, valores del intervalo $[0, 1]$ y distribuciones de probabilidad sobre los términos del lenguaje respectivamente.

Definamos ahora la noción de Signatura. Intuitivamente la signatura es el medio para especificar el lenguaje bajo estudio. Tiene un conjunto de símbolos (constantes y funciones) y una función de aridad como en la definición usual de signatura, pero con la novedad consideremos además una función que especifica la naturaleza probabilística de cada símbolo de función. Es decir:

Definición 3.1 *Una signatura Σ será una terna $\Sigma = (F, r, \Delta)$ donde F es un conjunto de símbolos de función, r una función que asigna a cada símbolo un número natural representando la aridad de dicho símbolo (los símbolos de aridad 0 son las constantes de la signatura), y $\Delta \in F \to \wp(pvar)$ es una función que asocia a cada símbolo de función $f \in F$ un conjunto de variables que sólo pueden tomar valores probabilísticos. Estos valores son entendidos como las probabilidades con las que es aplicado el símbolo f y está definida por: $\Delta(f) = c$ sii $\#c = r(f)$ y $\sum_{p \in c} p = 1$.*

Se definen ahora los elementos que puede generar una signatura como es usual:

Definición 3.2 *Dado el conjunto de variables var y $V \subseteq var$, definimos los términos de Σ con respecto a V $(T(\Sigma, V))$[5] inductivamente como:*

$$\begin{aligned} T_o(\Sigma, V) &= V \\ T_{k+1}(\Sigma, V) &= T_k(\Sigma, V) \cup \{f(t_1, .., t_n) / f \in F \ , \ t_i \in T_k(\Sigma, V), r(f) = n\} \\ T(\Sigma, V) &= \bigcup_{k \geqslant 0} T_k(\Sigma, V) \end{aligned}$$

Por último, demos forma a esta estructura sintáctica definiendo los sistemas de especificación de transiciones probabilísticos concurrentes rotulados. Esta estructura es un sistema de pruebas a lo [GV92] y se usan para especificar a los $SPCR$.[6] Están compuestos de una signatura (que especifica al lenguaje subyacente), un conjunto de rótulos (igual que en los $SPCR$) y un conjunto de reglas de inferencia que le dan semántica operacional al lenguaje.

Definición 3.3 *Un sistema de especificación de transiciones probabilísticas concurrentes rotuladas con acciones $(SEPCR)$ es una terna (Σ, A, R) donde Σ es una signatura, A es un conjunto de símbolos que representan las posibles acciones sobre el sistema y R es un conjunto de reglas de inferencia de la forma:*

$$\frac{\{t_i \xrightarrow{a_i} d_i, i \in I\}}{t \xrightarrow{a} d} \Bigg[D$$

con $t_i, t \in \mathbf{T}(\Sigma)$, $a_i, a \in A$, $d, d_i \in \mu var$, I es un conjunto índice y por último D es un conjunto de condiciones de aplicabilidad de la regla. Si en una regla el antecedente es un conjunto vacío, ésta será llamada un axioma. Los elementos de la forma $t \xrightarrow{a} d$ serán llamados transiciones.

Ejemplo 3.1 **($P_p^{\oplus}$)** *Definición de un SEPCR que especifica un lenguaje de procesos probabilístico. Supongamos $\mathcal{A} = \{a, b, c\}$ un conjunto de acciones. Sea $\Sigma_p^{\oplus}$ tal que contiene un símbolo de constante por cada símbolo de $\mathcal{A}$ y dos constantes δ, ϵ representando la inacción y la acción inmediatamente exitosa respectivamente. Supongamos además que $\Sigma_p^{\oplus}$ contiene además tres símbolos de función " + " (representa la composición alternativa), "." (representando la composición secuencial) y "$\oplus$" (representando la composición alternativa probabilística), todas funciones binarias.[7] Como conjunto A de rótulos usamos $A = \mathcal{A}_{\surd}$ donde $\mathcal{A}_{\surd} = \mathcal{A} \cup \{\surd\}$ con $\surd$ un símbolo especial usado para denotar la acción de terminación exitosa. Definimos $P_p^{\oplus}$ como:*

$$P_p^{\oplus} = (\Sigma_p^{\oplus}, \mathcal{A}_{\surd}, R_p^{\oplus})$$

[5] Como convención notacional tomaremos $\mathbf{T}(\mathbf{\Sigma}) = T(\mathbf{\Sigma}, var)$ y $T(\mathbf{\Sigma}) = T(\mathbf{\Sigma}, \varnothing)$.

[6] Los sist. de esp. de trans. prob. concurrentes rotulados como especifican una semántica del lenguaje, son considerados ellos mismos una semántica operacional del lenguaje.

[7] Quienes estén familiarizados con las álgebras de procesos podrán apreciar la semejanza con la signatura de $BPA_{\delta,\epsilon}^{\oplus}$. Ver [GV92]

Donde $R_p^{\oplus}$ *es:*

1. $a \xrightarrow{a} \{(\epsilon, 1)\}\ a \neq \surd$ 2. $\epsilon \xrightarrow{\surd} \{(\delta, 1)\}$ 3. $\frac{t_1 \xrightarrow{a} d}{t_1 + t_2 \xrightarrow{a} d}$ 4. $\frac{t_2 \xrightarrow{a} d}{t_1 + t_2 \xrightarrow{a} d}$

5. $\frac{t_1 \xrightarrow{a} \{(s_i, p_i), i \in [1..n]\}}{t_1 . t_2 \xrightarrow{a} \{(s_i . t_2, p_i), i \in [1..n]\}}$ 6. $\frac{t_1 \xrightarrow{\surd} d_1, t_2 \xrightarrow{a} d_2}{t_1 . t_2 \xrightarrow{a} d_2}$ 7. $\frac{t_1 \xrightarrow{a} d_1, t_2 \xrightarrow{a} d_2}{t_1 \oplus t_2 \xrightarrow{a} p_1 d_1 + p_2 d_2} \Big[\Delta(\oplus) = \{p_1, p_2\}$

□

Dado que en las reglas pueden aparecer variables y que en los $SPCR$ los estados son términos cerrados, necesitamos la noción de sustituir una variable por un término cerrado, basados en la definición clásica de sustitución ground, la extenderemos a términos y transiciones como:

Definición 3.4 *Un entorno de aplicación de una regla o instanciación de una regla* σ *es un mapeo* $\sigma = \sigma_t \cup \sigma_p$ *con* $\sigma_t \in var \to T(\Sigma), y\ \sigma_p \in pvar \to [0, 1]$. *Definimos además una extensión de la noción de entorno para poder ser aplicada sobre términos e incluso sobre transiciones y distribuciones como sigue:*
$\sigma(f(t_1, .., t_n)) = f(\sigma(t_1), .., \sigma(t_n))$ *si* $r(f) = n \geq 1$
$\sigma(f) = f$ *si* $r(f) = 0$
$\sigma(\{(t_i, p_i)\}) = \{(\sigma(t_i), \sigma(p_i))\}$
$\sigma(t \xrightarrow{a} \mu) = \sigma(t) \xrightarrow{a} \sigma(\mu)$

A continuación, se define el mecanismo de prueba. Este es al estilo de [GV92] y se basa en la posibilidad de aplicar las reglas disponibles siempre y cuando se encuentren entornos conformes que permitan tal aplicación.

Definición 3.5 *Sea* $P = (\Sigma, A, R)$ *un* $SEPCR$. *Una prueba de una transición* φ *en* P *es un árbol cuyos nodos son rotulados con transiciones* $t \xrightarrow{a} \mu$ *con* $t \in T(\Sigma)$, $a \in A$, $\mu \in Distr(T(\Sigma))$, *tal que el rótulo de la raíz es* φ *y si* $\varkappa$ *es el rótulo de un nodo* q *y* $\varkappa_i, i \in I$ *los rótulos de los nodos inmediatamente anteriores a* q, *entonces hay una regla*

$$\frac{\{\phi_i, i \in I\}}{\phi} \Big[D$$

en R *y un entorno* σ *tal que* $\sigma(\phi_i) = \varkappa_i, \sigma(\phi) = \varkappa$ *y* σ *hace válidas las condiciones de aplicabilidad en* D. *En el caso en que este árbol exista diremos que* $P \vdash \varphi$.

Por último, se introduce la idea de la "menor prueba", la cual será útil para las demostraciones por inducción:

Definición 3.6 *Sea* P *un* $SEPCR$ *y sean* A_i *con* $1 \leq i \leq n$ *los árboles de prueba de la transición* φ *en* P, *sean además* h_i *las alturas de estos árboles, si* $k = min(\{h_i\})$ *diremos entones que* $P \vdash_k \varphi$.

3.2. Interpretación

Se define ahora de qué forma los $SEPCR$ especifican a los $SPCR$, se le dará semántica a los $SEPCR$ en función de los $SPCR$ al estilo de [GV92].

La idea intuitiva detrás de la especificación es que una transición probable en el $SEPCR$ se vea reflejada en el $SPCR$. Al revés también, es decir, que una transición en el $SPCR$ sea probable en el $SEPCR$. [8] Es claro que los estados del $SPCR$ deberán ser términos de Σ y que la transición entre estados del $SPCR$ tendrá que ver con las pruebas posibles en el $SEPCR$. Para ser más precisos se pedirá que una transición se pueda hacer en el $SPCR$ si y sólo si se puede probar en el $SEPCR$.

Definición 3.7 *Sea* $P = (\Sigma, A, R)$ *un* $SEPCR$ *entonces definimos el* $SPCR$ *especificado por* P *como* $SPCR(P) = (T(\Sigma), A, Steps)$ *donde* $t \xrightarrow{a} \mu$ *en* $SPCR(P)$ *sii* $P \vdash t \xrightarrow{a} \mu$ *o equivalentemente*

$$(t, (a, \mu)) \in Steps \text{ sii } P \vdash t \xrightarrow{a} \mu$$

[8] Para aquellos familiarizados con las lógicas, sería como pedirle que el sistema de reglas sea correcto y completo.

Ejemplo 3.2 *En $P_p^{\oplus}$ (ej. 3.1, pág. 4), se puede verificar que el árbol mostrado en la figura 2 constituye una prueba de la transición $(\epsilon.(a+b)).c \xrightarrow{a} \{(\epsilon.c, 1)\}$, y que el sistema de transición asociado al término $(\epsilon.(a+b)).c$, el cual forma parte de $TS(P_p^{\oplus})$, es el mostrado en la figura 3.*

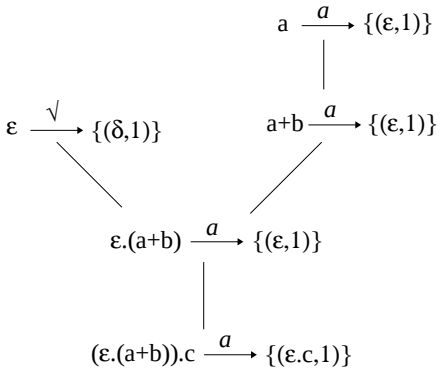


Figura 2: Prueba de ejemplo en $P_p^{\oplus}$

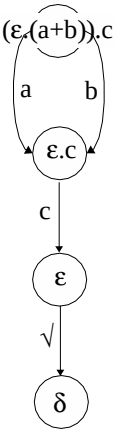


Figura 3: Parte de $TS(P_p^{\oplus})$

□

4. Formatos de regla propuestos.

Las constantes de Σ son especificadas por axiomas que directamente describen su proceder. Podemos proponer un formato inicial para tales axiomas: **Formato 0**

$$\frac{}{f \xrightarrow{a} \mu}$$

con $f \in F$, $r(f) = 0$ y $\mu \in Distr(T(\Sigma))$. Estos axiomas no son el interés de este estudio. En este trabajo se proponen cinco formatos de regla que aseguran que la bisimulación fuerte es una congruencia con respecto a los símbolos de función de la signatura, se presentan en orden evolutivo, donde la diferencia entre uno y otro manifiesta la intensión de relajar alguna restricción para obtener un formato más general, estos son:

Formato 1

$$\frac{\{t_i \xrightarrow{a} d_i, i \in [1..n]\}}{f(t_1,..,t_n) \xrightarrow{a} \sum_{i=1}^{n} p_i(d_i)} \Bigg[\Delta(f) = \{p_1,..,p_n\}$$

con $t_i \in var$, $f \in F$, $a \in A$, $d_i \in \mu var$, $p_i \in pvar$.

Donde $\sum_{i=1}^{n} p_i(d_i)$ es la combinación convexa de $\{d_i, i \in [1..n]\}$ con respecto a $\{p_i, i \in [1..n]\}$[9]. Intuitivamente esta regla permite especificar funciones cuyo proceder depende explícitamente del proceder de todos sus operandos y además depende de que éstos ejecuten la misma acción.

Formato 2

$$\frac{\{t_i \xrightarrow{a} \{(x_i^l, p_i^l), l \in L_{t_i}\}, \in [1..n]\}}{f(t_1,..,t_n) \xrightarrow{a} \sum_{i=1}^{n} q_i(\{(t[x : x_i^l], p_i^l), l \in L_{t_i}\})} \Bigg[\Delta(f) = \{q_1,..,q_n\}$$

con $x_i^l, t_i \in var$, $f \in F$, $a \in A, q_i, p_i^l \in pvar, t \in T(\Sigma, \{x\}), L_{t_i}$ son n conjuntos índice, uno para cada t_i.

[9] Para más detalle sobre la combinación convexa ver [Fiuri].

Donde $t[x:y]$ es el término que resulta de sustituir todas las ocurrencias de x en t por una de y. Intuitivamente esta regla permite especificar funciones cuyo proceder dependa implícitamente del proceder de todos sus operandos y además depende de que éstos ejecuten la misma acción. La dependencia implícita es a través de la sustitución sobre un término de una variable.

Formato 3

$$\frac{\{t_i \xrightarrow{a_i} d_i, i \in [1..n]\}}{f(t_1,..,t_n) \overset{g(a_1,..,a_n)}{\rightarrow} \sum_{i=1}^{n} p_i(d_i)} \left[\Delta(f) = \{p_1,..,p_n\} \right.$$

con $t_i \in var$, $f \in F$, $a_i \in A, g \in A^n \rightarrow A$, $d_i \in \mu var$, $p_i \in pvar$. Intuitivamente este formato permite especificar funciones cuyo proceder depende directamente de sus operandos pero estos pueden ejecutar diferentes acciones.

Formato 4

$$\frac{\{t_i \xrightarrow{a_i} d_i, i \in [1..n]\}}{f(s_1,..,s_m) \overset{g(a_1,..,a_n)}{\rightarrow} \sum_{i=1}^{n} p_i(d_i)} \left[\Delta(f) = \{p_1,..,p_n\} \right.$$

con $t_i, t'_k \in var$, $f \in F$, $a_i \in A, g \in A^n \rightarrow A$, $d_i \in \mu var$, $p_i \in pvar$, $r(f) = m \geq n \neq 0$ *y* $s_j = t_i$ *para algún* $i \in [1..n]$ *o* $s_j = t'_k$ *para algún* $k \in [1..m-n]$.

Intuitivamente este formato permite especificar las funciones cuyo proceder depende explícitamente sólo de aquellos operandos cuyo comportamiento es conocido.

Formato 5

$$\frac{\{t_i \xrightarrow{a_i} d_i, i \in [1..n]\}}{f(s_1,..,s_m) \overset{g(a_1,..,a_n)}{\rightarrow} \breve{f}(e_1,..,e_m)}$$

con $t_i, t'_k \in var$, $f \in F$, $a_i \in A, g \in A^n \rightarrow A$,$d_i \in \mu var$, $s_j \in \{t_i, t'_k\}$ *para* $k \in [1..(m-n)]$, $n \neq 0$ *y por último* $e_j = d_i$ *si* $s_j = t_i$ *o* $e_j = t'_k$ *si* $s_j = t'_k$.

Donde $\breve{f}$ es una función mixta. [10] Intuitivamente este formato permite especificar las funciones cuyo proceder depende de aquellos operandos con comportamiento conocido, como asi también desconocido.

5. Bisimulación como congruencia.

Una propiedad interesante de una relación de equivalencia es que sea una congruencia con respecto a alguna operación.[11] Intuitivamente una congruencia con respecto a f (de aridad n) satisface que si n términos t_i están relacionados a otros n términos s_i entonces $f(t_1,..,t_n)$ está relacionado a $f(s_1,..,s_n)$. Tomemos la definicion clásica de congruencia:

Definición 5.1 (Congruencia) *Sea* $\Sigma = (F, r, \Delta)$ *una signatura y sea* $r(f) = n$ *entonces una relación de equivalencia* $\otimes$ *sobre los términos de* Σ *será llamada una congruencia si:*

$$(\forall f \in F : t_i \otimes s_i \Rightarrow f(t_1,..,t_n) \otimes f(s_1,..,s_n))$$

Se presenta ahora el resultado central al desarrollo de este trabajo. Es un teorema semejante al de [GV92], pero para el caso probabilístico y para los formatos propuestos. Se incluye también la demostración, sólo para el primer formato de regla propuesto. [12]

Teorema 1 *Sea* $P = (\Sigma, A, R)$ *un SEPCR, donde las reglas en R tienen alguno de los formatos propuestos, entonces la relación de bisimulación fuerte sobre el SPCR especificado por P es una congruencia con respecto a los símbolos de función de* Σ*, más precisamente:*

$$(\forall f \in F : (\forall i : u_i \underline{\leftrightarrow} v_i) \Rightarrow (f(u_1,..,u_n) \underline{\leftrightarrow} f(v_1,..,v_n)))$$

[10] Ver apéndice A.
[11] Entre otras, asegura la composicionalidad de la sustitucion contextual.
[12] Para ver todas las demostraciones consultar [Fiuri].

La estrategia para demostrar el teorema se basa en la de Groote y Vaandrager [GV92]. Consta en definir una relación que sea una congruencia y que contenga a la bisimulación fuerte. Luego se demuestra que en realidad esta relación también es una bisimulación por lo que entonces deberá estar contenida en la bisimulación fuerte y en consecuencia coincidir con ésta.

La parte central de la demostración se hace por inducción en la altura de la prueba de una transición.

Prueba del Teorema 1 para el caso del primer formato.

Sea $\mathcal{R} \subseteq S \times S$ con $S = T(\Sigma)$ la mínima relación de equivalencia que satisface:

a) La bisimulación fuerte está incluida en $\mathcal{R}$ o sea que:

$$\forall s,t \in T(\Sigma) : s \underline{\leftrightarrow} t \Rightarrow s\mathcal{R}t$$

b) $\mathcal{R}$ es una congruencia o sea que:

$$\forall f \in F : (\forall u_i v_i \in T(\Sigma) : 1 \leq i \leq r(f) : u_i \mathcal{R} v_i) \Rightarrow f(u_1,..,u_n)\mathcal{R}f(v_1,..,v_n)$$

Probaremos que $\mathcal{R}$ es una bisimulación, o sea que $\mathcal{R} \subseteq \underline{\leftrightarrow}$. Luego por definición $\underline{\leftrightarrow} \subseteq \mathcal{R}$. Por consiguiente $\underline{\leftrightarrow} = \mathcal{R}$ o sea que $\mathcal{R}$ es una congruencia.

Sea $(u,v) \in \mathcal{R}$, tenemos dos casos que analizaremos por separado:

Caso a: supongamos que $(u,v) \in \mathcal{R}$ por satisfacer la propiedad a) de la definición de $\mathcal{R}$, es decir que $(u,v) \in \underline{\leftrightarrow}$. Entonces por la definición de bisimilaridad (def. 2.4, pág. 3), sabemos que satisfacen:

$$u \xrightarrow{a} \mu \Rightarrow (\exists \eta \in Distr(T(\Sigma)) : (v \xrightarrow{a} \eta \ y \ (\forall c \in T(\Sigma)/\underline{\leftrightarrow} : \mu[c] = \eta[c]))) \qquad \text{pf-A}$$

Además, como $\mathcal{R}$ es la menor relación de equivalencia que contiene a $\underline{\leftrightarrow}$ sabemos (de cálculo de relaciones) que :

$$\forall c \in T(\Sigma)/\mathcal{R} : \exists c_{i_1},..,c_{i_n} \in T(\Sigma)/\underline{\leftrightarrow} : c = \bigcup_{j=1}^{n} c_{i_j} \qquad \text{pf-B}$$

Para que $\mathcal{R} \subseteq \underline{\leftrightarrow}$, $\mathcal{R}$ debe ser una bisimulación, o sea que debe satisfacer la condición de bisimilitud para u y v:

$$u \xrightarrow{a} \mu \Rightarrow \exists \eta \in Distr(T(\Sigma)) : (v \xrightarrow{a} \eta \ y \ (\forall c \in T(\Sigma)/\mathcal{R} : \mu[c] = \eta[c]))$$

sii (de las propiedades de relaciones, i.e. por pf-B)

$$u \xrightarrow{a} \mu \Rightarrow \exists \eta \in Distr(T(\Sigma)) : (v \xrightarrow{a} \eta \ y \ (\forall c \in T(\Sigma)/\mathcal{R} : \mu[\bigcup_{j=1}^{n} c_{i_j}] = \eta[\bigcup_{j=1}^{n} c_{i_j}])))$$

sii (de la distributividad de las distribuciones de probabilidad sobre la unión disjunta[13])

$$u \xrightarrow{a} \mu \Rightarrow \exists \eta \in Distr(T(\Sigma)) : (v \xrightarrow{a} \eta \ y \ (\forall c \in T(\Sigma)/\mathcal{R} : \sum_{j=1}^{n} \mu[c_{i_j}] = \sum_{j=1}^{n} \eta[c_{i_j}])))$$

lo cual se satisface ya que $u \underline{\leftrightarrow} v$ i.e. por pf-A

Caso b: Sea ahora el caso en que $(u,v) \in \mathcal{R}$ por satisfacer la propiedad b) de la definición de $\mathcal{R}$, es decir que hay $f \in F, u_i \in T(\Sigma), v_i \in T(\Sigma)$ con $i \in [1..n]$ tal que $u_i \mathcal{R} v_i$ y $u = f(u_1,..,u_n)$ y $v = f(v_1,..,v_n)$.

Para que $\mathcal{R} \subseteq \underline{\leftrightarrow}$, $\mathcal{R}$ debe ser una bisimulación, o sea que debe satisfacer la condición de bisimilitud (def. 2.4, pág. 3) para u y v:

$$u \xrightarrow{a} \rho \Rightarrow (\exists \eta \in Distr(T(\Sigma)) : v \xrightarrow{a} \eta \ y \ (\forall c \in T(\Sigma)/\mathcal{R} : \rho[c] = \eta[c]))$$

o lo que es lo mismo

$$f(u_1,..,u_n) \xrightarrow{a} \rho \Rightarrow (\exists \eta \in Distr(T(\Sigma)) : f(v_1,..,v_n) \xrightarrow{a} \eta \ y \ (\forall c \in T(\Sigma)/\mathcal{R} : \rho[c] = \eta[c]))$$

[13] $\mu[A \cup B] = \mu[A] + \mu[B] + \mu[A \cap B]$

ahora bien, de la semánica de nuestra estructura sintáctica (def. 3.7, pág. 5), tenemos que $f(u_1,..,u_n) \xrightarrow{a} \rho$ sii $P \vdash f(u_1,..,u_n) \xrightarrow{a} \rho$ y $f(v_1,..,v_n) \xrightarrow{a} \eta$ sii $P \vdash f(v_1,..,v_n) \xrightarrow{a} \eta$, por lo que la condición de bisimilitud queda entonces:

$$P \vdash f(u_1,..,u_n) \xrightarrow{a} \rho \Rightarrow (\exists \eta \in Distr(T(\Sigma)) : P \vdash f(v_1,..,v_n) \xrightarrow{a} \eta \; y \; (\forall c \in T(\Sigma)/\mathcal{R} : \rho[c] = \eta[c]))$$

Lo probaremos por inducción en la altura del menor árbol de prueba de $P \vdash f(u_1,..,u_n) \xrightarrow{a} \rho$, supongamos entonces que: $P \vdash f(u_1,..,u_n) \xrightarrow{a} \rho$.

Caso base: el árbol de prueba de $P \vdash f(u_1,..,u_n) \xrightarrow{a} \rho$ tiene altura 0 es decir que la prueba consiste de la aplicación de un axioma. Supongamos que se aplica $\frac{}{f \xrightarrow{a} \mu}$ (el cual se corresponde con el formato 0). Pero como $n = 0 = r(f)$ por ser una constante, $f(u_1,..,u_n) = f(v_1,..,v_n) = f$. El resultado se satisface trivialmente ya que se usa el mismo axioma para la prueba de $f(v_1,..,v_n)$.

Hipótesis Inductiva: sea $s,t \in T(\Sigma)$ con $s\mathcal{R}t, \gamma \in Distr(T(\Sigma))$ y $a \in A$ entonces

$$\forall j \leq k : (P \vdash_j s \xrightarrow{a} \gamma \Rightarrow (\exists \delta \in Distr(T(\Sigma)) : P \vdash t \xrightarrow{a} \delta \; y \; (\forall c \in T(\Sigma)/\mathcal{R} : \gamma[c] = \delta[c]))) \qquad \text{(pf-HI)}$$

Paso Inductivo: sean ahora u y v como antes. Supongamos que $P \vdash_{k+1} f(u_1,..,u_n) \xrightarrow{a} \rho$ y supongamos además que la última regla usada es del formato propuesto, siendo instanciada con un entorno σ tal que:

$$\sigma(x) = \begin{cases} u_i \; si \; x = t_i \\ \mu_i \; si \; x = d_i \\ \pi_i \; si \; x = p_i \end{cases} \qquad \frac{\{u_i \xrightarrow{a} \mu_i, i \in [1..n]\}}{f(u_1,..,u_n) \xrightarrow{a} \sum_{i=1}^{n} \pi_i(\mu_i)}$$

considerando además que las condiciones de aplicabilidad son válidas, la regla instanciada queda entonces como la mostrada arriba, o sea que $\rho = \sum_{i=1}^{n} \pi_i(\mu_i)$. De esta manera tenemos que para cada i, $u_i \xrightarrow{a} \mu_i$ es probable en P por un árbol de altura menor o igual a k. Como $u_i \mathcal{R} v_i$ podemos aplicar la hipótesis inductiva (pf-HI), tenemos entonces que:

$$\forall i \in [1..n] : (\exists \eta_i \in Distr(T(\Sigma)) : P \vdash v_i \xrightarrow{a} \eta_i \; y \; (\forall c \in T(\Sigma)/\mathcal{R} : \mu_i[c] = \eta_i[c])) \qquad \text{(pf-hi)}$$

Podemos entonces aplicar la misma regla que antes, con una sustitución conforme, pero esta vez sobre las pruebas de $P \vdash v_i \xrightarrow{a} \eta_i$ y obtener una prueba de $P \vdash f(v_1,..,v_n) \xrightarrow{a} \eta$.

Sea entonces σ' un entorno tal que:

$$\sigma'(x) = \begin{cases} v_i \; si \; x = t_i \\ \eta_i \; si \; x = d_i \\ \sigma(x) \qquad c \; c \end{cases} \qquad \frac{\{v_i \xrightarrow{a} \eta_i, i \in [1..n]\}}{f(v_1,..,v_n) \xrightarrow{a} \sum_{i=1}^{n} \pi_i(\eta_i)}$$

la regla instanciada con σ' queda entonces como la mostrada arriba.

Notar que las condiciones de aplicabilidad son válidas (tomando las mismas probabilidades para la aplicación de f), por lo que es válida la aplicación de la regla. Además implícitamente se estudia el caso en el que la función f se aplica con las mismas probabilidades en ambos casos. Esto es tanto a los $u_i's$ como a los $v_i's$ se les aplica f con probabilidades $\{\pi_1,..,\pi_n\}$.

Probamos entonces que hay una prueba de $P \vdash f(v_1,..,v_n) \xrightarrow{a} \eta$, con $\eta = \sum_{i=1}^{n} \pi_i(\eta_i)$.

Queda por último probar que $(\forall c \in T(\Sigma)/\mathcal{R} : \rho[c] = \eta[c])$, para ello si $c_o \in T(\Sigma)/\mathcal{R}$

$$\rho[c_o] = (\sum_{i=1}^{n} \pi_i(\mu_i))[c_o] = \sum_{i=1}^{n} \pi_i(\mu_i[c_o]) = \sum_{i=1}^{n} \pi_i(\eta_i[c_o]) = (\sum_{i=1}^{n} \pi_i(\eta_i))[c_o] = \eta[c_o]$$

las igualdades son válidas de la distributividad de la combinación convexa con respecto a la aplicación sobre conjuntos y de la hipótesis inductiva (pf-hi). Luego como c_o es arbitrario entonces el resultado es válido $\forall c \in T(\Sigma)/\mathcal{R}$. Hemos probado así que $\forall s,t \in T(\Sigma)$:

$$s\mathcal{R}t \Rightarrow (P \vdash s \xrightarrow{a} \gamma \Rightarrow (\exists \delta \in Distr(T(\Sigma)) : P \vdash t \xrightarrow{a} \delta \; y \; (\forall c \in T(\Sigma)/\mathcal{R} : \gamma[c] = \delta[c])))$$

Es decir que $\mathcal{R}$ es una bisimulación (def. 2.4, pág. 3) y por consiguiente $\mathcal{R} \subseteq \underline{\leftrightarrow}$. Dado que $\underline{\leftrightarrow} \subseteq \mathcal{R}$ por definición de $\mathcal{R}$, tenemos que $\mathcal{R} = \underline{\leftrightarrow}$. Por lo que la bisimulación fuerte cumple también la propiedad b) de la definición de $\mathcal{R}$, en otras palabras es una congruencia.

■

6. Expresividad

Se presenta ahora un breve análisis de expresividad, definiendo las reglas que especifican a algunos de los operadores conocidos en la literatura de las álgebras de procesos.[14]

6.1. *SEPCR* con *BPA* subyascente (*SBPA*).

Comencemos especificando el lenguaje de BPA.[15] El $SEPCR$ que especifica a BPA es el definido en el ejemplo 3.1, pág. 4. Enriquezcamos BPA con un operador de restricción, el cual restringe la ejecución de un proceso a la no ejecución de un conjunto de procesos atómicos. Si p es un proceso y $L \subseteq A$ entonces $p\backslash L$ será un proceso que se comporte como p mientras no ejecute alguna acción que esté en L.

La regla que especifica a $\backslash$ es:

$$\frac{t \xrightarrow{a} d}{t\backslash L \xrightarrow{a} d\breve{\backslash} L} \Big[a \notin L$$

donde la operación $\breve{\backslash} : Distr(T(\Sigma)) \times T(\Sigma) \to Distr(T(\Sigma))$ está definida por:

$\{(t_1, p_1), .., (t_n, p_n)\}\breve{\backslash}L = \{(t_1\backslash L, p_1), .., (t_n\backslash L, p_n)\}$.

Claramente se ve que $\breve{\backslash}$ es una función mixta (ver apéndice A) y que la regla tiene la forma propuesta por el quinto formato.

6.2. *SEPCR* con *CCS* subyacente (*SCCS*).

Nos concentremos en la extensión probabilística de CCS de Milner [Miln89]. En ese trabajo se implementa el paralelismo asincrónico a través de la comunicación entre procesos. Es decir los procesos se ejecutaran en forma independiente hasta que cierta condición de sincronización tome lugar y entonces en ese momento hacen una acción conjunta.[16] Si τ es la constante que representa un cómputo interno del sistema, el cual no es visible para el entorno. Si $\_ : A \to A$ implementa la noción de acción atómica complementaria, la cual permite implementar la sincronización entre procesos ya que dos procesos estarán sincronizados cuando puedan ejecutar acciones complementarias. Y si además el resto de símbolos es como en el caso anterior, las reglas para especificar este operador son:

$$\frac{t \xrightarrow{a} d}{\delta \| t \xrightarrow{a} d} \quad \frac{t \xrightarrow{a} d}{t \| \delta \xrightarrow{a} d} \quad \frac{t_2 \xrightarrow{b} d_2}{t_1 \| t_2 \xrightarrow{b} t_1 \breve{\|} d_2} \quad \frac{t_1 \xrightarrow{a} d_1}{t_1 \| t_2 \xrightarrow{a} d_1 \breve{\|} t_2} \quad \frac{\{t_1 \xrightarrow{a} d_1, t_2 \xrightarrow{b} d_2\}}{t_1 \| t_2 \xrightarrow{\tau} d_1 \breve{\|} d_2} \Big[b = \overline{a}$$

donde las operaciones $\breve{\|}$ están definidas por[17]:

$\breve{\|} : Distr(T(\Sigma)) \times T(\Sigma) \to Distr(T(\Sigma))$ está definida por:

$\{(t_1, p_1), .., (t_n, p_n)\} \breve{\|} t = \{(t_1 \| t, p_1), .., (t_n \| t, p_n)\}$

$\breve{\|} : T(\Sigma) \times Distr(T(\Sigma)) \to Distr(T(\Sigma))$ está definida por:

$t \breve{\|} \{(t_1, p_1), .., (t_n, p_n)\} = \{(t \| t_1, p_1), .., (t \| t_n, p_n)\}$

$\breve{\|} : Distr(T(\Sigma)) \times Distr(T(\Sigma)) \to Distr(T(\Sigma))$ está definida por:

$$\{(t_1, p_1), .., (t_n, p_n)\} \breve{\|} \{(s_1, q_1), .., (s_m, q_m)\} = \{(t_1 \| s_1, p_1 q_1), .., (t_1 \| s_m, p_1 q_m),$$
$$\vdots \qquad\qquad \vdots$$
$$(t_n \| s_1, p_n q_1), .., (t_n \| s_m, p_n q_m)\}$$

Estas reglas se ajustan a los formatos quinto y primero.

[14] Para un análisis más detallado consultar [Fiuri].

[15] Más precisamente $BPA^{\oplus}_{\delta,\epsilon}$ a lo [GV92].

[16] Para más detalle consultar [Baier, Cap.4].

[17] Notar que esta sobrecarga de operadores es resuelta por el tipo de los operandos y que $\breve{\|}$ son 3 funciones mixtas.

6.3. $SEPCR$ con $SCCS$ subyacente ($SSCCS$).

Enfoquémonos ahora en la extensión probabilística de $SCCS$ de Milner [Miln83]. En ese trabajo se implementa una forma de paralelismo sincrónico en donde los componentes de los procesos trabajan en forma dependiente del tiempo. Cada paso de la ejecución paralela está compuesta de exactamente una acción de cada componente.[18] En el operador de composición paralela sincrónica $\times$ todas las transiciones están compuestas de ejecuciones individuales de sus componentes. Si $* : A \times A \to A$ implementa la noción de ejecución conjunta de dos acciones, donde donde $a * b = c$ significa que el resultado de ejecutar simultáneamente a y b es c. Y si el resto de los símbolos es como antes, entonces las reglas que especifican a este operador son:

$$\frac{\{t_1 \xrightarrow{a} d_1, t_2 \xrightarrow{b} d_2\}}{t_1 \times t_2 \xrightarrow{c} d_1 \breve{\times} d_2} \Big[a * b = c \qquad \frac{}{\delta \times t \xrightarrow{a} \mu_\varepsilon} \qquad \frac{}{t \times \delta \xrightarrow{a} \mu_\varepsilon}$$

donde la operación $\breve{\times} : Distr(T(\Sigma)) \times Distr(T(\Sigma)) \to Distr(T(\Sigma))$ está definida por:

$$\{(t_1, p_1), .., (t_n, p_n)\} \breve{\times} \{(s_1, q_1), .., (s_m, q_m)\} = \{(t_1 \times s_1, p_1 q_1), .., (t_1 \times s_m, p_1 q_m),$$
$$\vdots \qquad\qquad \vdots$$
$$(t_n \times s_1, p_n q_1), .., (t_n \times s_m, p_n q_m)\}$$

Notar que $\breve{\times}$ es una función mixta donde no hay términos como parámetros, sólo distribuciones.

6.4. $SEPCR$ con $PACP$ subyacente ($SPACP$).

El último de los operadores de paralelismo analizados fue introducido por Baeten, Bergstra y Smolka en [BBS92], como una extensión de ACP [BeKl84], para contemplar casos probabilísticos. Es llamado composición paralela probabilística ($\|_{\pi,\theta}$). La diferencia con el operador asincrónico visto en CCS es que se resuelve el no determinismo entre la comunicación entre procesos y la ejecución autónoma de éstos, mediante probabilidades. Se introducen dos parámetros probabilísticos al operador. El primero es la probabilidad de que se produzca sincronización (si es que está disponible, y en tal caso es como el paralelismo asincrónico ya visto) y el segundo es la probabilidad con que se ejecutará la primera acción del primer operando si es que falla la sincronización. Para este operador necesitamos capturar la noción de hacer una transición consumiendo un símbolo (digamos a) de entrada, con cierta probabilidad (digamos π), es decir, las transiciones deberán ser de la forma: $t \overset{(\pi,a)}{\to} \mu$.[19] Este tipo de transiciones corresponden a un modelo semántico que no es el que está bajo estudio, una mezcla entre el simple y el reactivo. Por esta razón, este operador no es especificable con los formatos propuestos.

7. Conclusiones

El principal aporte de este trabajo es el de hacer una extensión probabilística de un formalismo bien conocido como el desarrollado en [GV92]. Se propone extender un resultado (bajo la forma de teorema). Para ello, en primer lugar se extiende la estructura sintáctica necesaria como para especificar los modelos probabilísticos subyacentes. En segundo lugar, se estudian condiciones que establezcan el resultado buscado (bajo la forma de reglas de inferencia de la estructura sintáctica). Por último, se construyen las pruebas pertinentes. No obstante hay un aporte más general e importante. Es el de proveer de una poderosa herramienta para aquellos que necesiten probar que la bisimulación es una congruencia cuando trabajan sobre modelos como los usados en este trabajo. No les hará falta hacer complejos y costosos cálculos propensos a errores. Les alcanzará con un chequeo sintáctico sobre las reglas que especifican a los operadores que manejan. Si estos se ajustan a los formatos propuestos aquí, tienen garantizada esta propiedad. Desde el punto de vista práctico, este resultado representa la posibilidad de poder cambiar un componente de un sistema por otro equivalente y mantener la equivalencia con el sistema original.

[18] Para más detalle consultar [Baier, Cap.4].
[19] Para mayor detalle consultar [Baier, Cap.4].

8. Apéndice

8.1. Apéndice A

Se introduce ahora una forma elegante de combinar distribuciones de probabilidad con términos del lenguaje. Como ya quedó expuesto, es útil a la hora de describir un operador en función de operandos cuyo funcionamiento se conoce, y operandos cuyo funcionamiento no se conoce. Sin perder la información acerca de éstos últimos y dando una distribución sobre $T(\Sigma)$.

Llamaremos a este tipo de funciones *funciones mixtas* y las definiremos así:

Definición 8.1 *Sean* $f \in F$, $\mu_1,..,\mu_n \in Distr(T(\Sigma))$ *y* $t'_1,..,t'_m \in T(\Sigma)$, *y sean* e_k *tales que* $e_k = \mu_i$ *para algún* $i \in [1..n]$ *o* $e_k = t'_j$ *para algún* $j \in [1..m]$. *Además sean* $D_k(T(\Sigma)) = Distr(T(\Sigma))$ *si* $e_k = \mu_i$ *para algún* $i \in [1..n]$ *o* $D_k(T(\Sigma)) = T(\Sigma)$ *si* $e_k = t'_j$ *para algún* $j \in [1..m]$.

Definimos: $\breve{f}(e_1,..,e_{n+m}) \in D_1(T(\Sigma)) \times \cdots \times D_{n+m}(T(\Sigma)) \to Distr(T(\Sigma))$ *de la siguiente manera:*

si $n = 0$ $\breve{f}(e_1,..,e_{n+m}) = \{(f(t_1,..,t_m),1)\}$

si $n \neq 0$ $\breve{f}(e_1,..,e_{n+m}) = \{(s,p)$ *donde:* $s = f(e'_1,..,e'_{n+m})$ *con* $e'_k = st(\mu_i)_j$ *si* $e_k = \mu_i$ *para algún* $i \in [1..n]$ *o* $e'_k = t'_l$ *si* $e_k = t'_l$ *para algún* $l \in [1..m]$ *y* $p = p_1..p_n$ *con* $p_i = pr(\mu_i)_j$, *para todo* j *válido.*$\}$

La demostración de que $\breve{f}$ es una distribución de probabilidades es dejada al lector interesado.[20]

Referencias

[BBS92] J. Baeten, J. Bergstra, S. Smolka: Axiomatizing Probabilistic Processes: ACP with Generative Probabilies, Proc. CONCUR'92, *Lecture Notes in Computer Science*, Vol. 630, pp 472-485, 1992.

[Baier] C. Baier: On Algorithmic Verification Methods for Probabilistic Systems, *Habilitationsschrift zur Erlangung der venia legendi der Fakultät für Mathematic und Informatic der Universität Mannheim*, 1998.

[BeKl84] J. Bergstra, J. Klop: Process Algebra for Synchronous Communication, *Information and Computation*, Vol. 60, pp 109-137, 1984.

[Fiuri] Ariel M. Fiuri: Sistemas de Especificación de Transiciones Probabilísticas: Formato de Reglas y Bisimulación como Congruencia, *Tesis de grado de la Licenciatura en Cs. de la Computación, Facultad de Matemática Astronomía y Física, Universidad Nacional de Córdoba,* 2004.

[GV92] J. F. Groote and F. Vaandrager: Structured Operational Semantics and Bisimulation as a Congruence, *Information and Computation 100:202-260,* 1992.

[LaSk89] K. Larsen, A. Skou: Bisimulation through Probabilistic Testing, Proc. POPL'89, 1989. Versión entera en *Information and Computation*, Vol. 94, pp 1-28, 1991.

[Miln83] R. Milner: Calculi for Synchrony and Asynchrony, *Theoretical Computer Science*, Vol. 25, pp 269-310, 1983.

[Miln89] R. Milner: Communication and Concurrency, Prentice Hall, 1989.

[Plot81] G. D. Plotkin: A Structural Approach to Operational Semantics, *Technical Report DAIMI FN-19, Computer Science Department, Aaurhus University,* 1981.

[20] Para ver la demostración, consultar [Fiuri].

An algebra for describing features

Pablo E. Martínez López Jerónimo Irazábal
LIFIA, Facultad de Informática, UNLP
CC.11 Correo Central
(1900) La Plata
Buenos Aires, Argentina
{fidel,jiraza}@sol.info.unlp.edu.ar

August 29, 2005

Abstract

Feature Diagrams (FD) are a graphical notation for describing all possible configurations of a software system focusing on the features that may differ in each of the configurations. In this paper we study an algebraic model which will allow us to define a textual language to describe features (FDL*); this language will be equivalent to feature diagrams in the sense that, for each FD we will be able to find a feature expression in FDL* that denote the same set of system configurations, and vice versa. We have chosen FDL*'s syntax to be equivalent to FDL's syntax (a known feature description language) because we think both have the same purpose; however we think that previous definition is not satisfactory and the goal of this paper is remedy this situation. We are interested in operating with feature expressions to be as simple as operating with numbers.

1 Introduction

A domain-specific language (DSL) provides a notation tailored towards an application domain and is based on the relevant concepts and features of that domain. As such, a DSL is a means to describe and generate members of a family of programs in the domain. A prerequisite for the design of a DSL is a detailed analysis and structuring of the application domain. Graphical feature diagrams have been proposed to organize the dependencies between such features, and to indicate which ones are common to all family members and which ones vary [5].

The purpose of this paper is to define a textual language to describe features equivalent to feature diagrams and also to have a formal way to operate over expression on this language. One of such a languages is FDL [5], but we do not agree with its definition because when we are operating feature expressions, lot of rules has to be applied in a specific order before and after being able to make one step of reduction, and it is not clear if the order in which expansion rules are applied does not matters; so we think this process is not flexible enough and makes the proof of properties too complicated. We would like to operate over feature expressions as we do with numbers. In order to do that we propose on this paper a new language FDL* based on algebraic grounds.

The plan of this paper is as follows. Section 2 briefly describe FDL [5] and describe the main disadvantages that it introduces. Then in Section 3 we introduce an Algebraic Model that captures the idea of describing features and then define a textual feature description language (FDL*), prove its properties and give the definition of FDL*, a new language for describing features in terms of this model and showed that any computation over FDL can be done with this language too. Notations and specifications are briefly explained in Section 4. Implementation issues are addressed in Section 5. An example is analyzed in detail in Section 6. Conclusions in Section 7 and the proof of selected proprositions in the Appendix complete the paper.

2 FDL definition and problems

The language FDL for describing features was proposed in [5]. FDL was designed with the goal of being a textual representation for feature diagrams, and its authors have proposed a way to manipulate feature expressions. But their approach is far of being appropriate to manipulate feature expressions. The following definitions are textually taken from [5] with the purpose of analyzing the original definition of FDL and to make this paper self contained.

Definition 2.0.1 (FDL)
An FDL definition consists of a number of feature definitions; each of such definitions consist on a feature name followed by ":" and a feature expression. A feature expression can consists of one of the following:

- an atomic feature,
- a composite feature: a named feature whose definition appears elsewhere,
- an optional feature: a feature expression followed by "?",
- mandatory features: a list of feature expressions enclosed in all( ),
- alternative features: a list of feature expressions enclosed in one-of( ),
- non-exclusive selection of features: a list of feature expressions enclosed in more-of( ),
- a default feature value: default = followed by an atomic feature

Example 2.0.1 (Car)

Car	: all(carBody, Transmission, Engine, HorsePower, pullsTrailer?)
Transmission	: one-of(automatic, manual)
Engine	: more-of(electric, gasoline)
HorsePower	: one-of(lowPower, mediumPower, highPower)

Notation 2.0.1
The following variables will be used throughout the rules that follow, restricting the nature of objects referred to the associates grown category.

Variable	*Type*
F	FeatureExpression
Fs	{ FeatureExpression "," }*
Ft	{ FeatureExpression "," }$^+$
A	AtomicFeature

Definition 2.0.2 (Normalization rules)
Normalization rules are used to simplify a given feature expression by eliminating duplicate features and degenerate cases of the various constructors.

[N1]	Fs, F, Fs′, F?, Fs″	=	Fs, F, Fs′, Fs″
[N2]	Fs, F, Fs′, F, Fs″	=	Fs, F, Fs′, Fs″
[N3]	F??	=	F?
[N4]	all(F)	=	F
[N5]	all(Fs, all(Ft), Fs′)	=	all(Fs, Ft, Fs′)
[N6]	one-of(F)	=	F
[N7]	one-of(Fs, one-of(Ft), Fs′)	=	one-of(Fs, Ft, Fs′)
[N8]	one-of(Fs, F?, Fs′)	=	one-of(Fs, F, Fs′)?
[N9]	more-of(F)	=	F
[N10]	more-of(Fs, more-of(Ft), Fs′)	=	more-of(Fs, Ft, Fs′)
[N11]	more-of(Fs, F?, Fs′)	=	more-of(Fs, F, Fs′)?
[N12]	default = A	=	A

Informaly speaking, **N1** combines mandatory and optional features in a list, **N2** removes duplicates in a list, **N3** joins duplicate optionals, **N4-N5** normalize special cases of all (nested alls are flattened), **N6-N7** normalize special cases of one-of (nested one-ofs are flattened), **N8** transforms a one-of containing one optional feature into an optional one-of, **N9-N10** normalize special cases of more-of (nested more-ofs are flattened), **N11** transforms a more-of containing one optional feature into an optional more-of, **N12** eliminates the default = annotation.

Normalized feature expression for Car is:

```
all (carBody,
   one-of(automatic, manual),
   more-of(electric, gasoline),
   one-of(lowPower, mediumPower, highPower),
   pullsTrailer?)
```

Definition 2.0.3 (Disjuntive normal form)
A feature expression fe is in disjunctive normal form if fe has the form:

$$\text{one-of}\begin{pmatrix} \text{all}(A_{1_{1_1}}, \ldots, A_{1_{n_1}}) \\ \vdots \\ \text{all}(A_{m_{m_1}}, \ldots, A_{m_{m_n}}) \end{pmatrix}$$

The outermost operator of a disjunctive normal form is a one-of, and its arguments are alls with only atomic features as arguments. The resulting representation is essentially a list of all possible configurations.

Expansion rules are used to transform a normalized feature expression into a disjunctive normal form.

Definition 2.0.4 (Expansion rules)

$$\begin{array}{lll}
\text{[E1]} & \text{all(Fs, F?, Ft)} & = \text{one-of}\begin{pmatrix}\text{all(Fs, F, Ft)}\\ \text{all(Fs, Ft)}\end{pmatrix} \\
\text{[E2]} & \text{all(Ft, F?, Fs)} & = \text{one-of}\begin{pmatrix}\text{all(Ft, F, Fs)}\\ \text{all(Ft, Fs)}\end{pmatrix} \\
\text{[E3]} & \text{all}\begin{pmatrix}\text{Fs}\\ \text{one-of(F, Ft)}\\ \text{Fs}'\end{pmatrix} & = \text{one-of}\begin{pmatrix}\text{all(Fs, F, Fs}')\\ \text{all(Fs, one-of(Ft), Fs}')\end{pmatrix} \\
\text{[E4]} & \text{all}\begin{pmatrix}\text{Fs}\\ \text{more-of(F, Ft)}\\ \text{Fs}'\end{pmatrix} & = \text{one-of}\begin{pmatrix}\text{all(Fs, F, Fs}')\\ \text{all(Fs, F, more-of(Ft),Fs}')\\ \text{all(Fs, more-of(Ft), Fs}')\end{pmatrix}
\end{array}$$

Informaly speaking, **E1,E2** translates an all containing an optional feature expression in two cases: one with and one without the feature, **E3** translates an all containing a one-of in two cases: one with the first alternative and one with the one-of with the first alternative removed, **E4** translates an all containing a more-of into three cases: one with the first alternative,one with the first alternative and the remaining more-of, and one with only the remaining more-of.

The expanded feature expression for Car is given in figure 1.

We have found several important problems in the previous definitions. They impact on readability and extensibily of the language and its implementation. The most relevant ones are:

- **Difficult to prove properties**
  Proving properties of the language FDL with this approach is difficult, if not impossible. We think that FDL should be more than a purely syntactic construction. In the next section we present a lot of properties of our language and make a proof of nontrivials ones.

- **Order of reduction matters**
  Is necessary to apply normalization rules from definition 2 before and after each step of reduction; moreover, the normalization rules have to be applied in a specific order. We would expect that two different reductions of the same feature expression arrive to equivalent feature expressions, independent of the order of application.
- **Not every feature expression can be reduced to disjunctive normal form**
  This happens when the outermost operator is not an all. For example: more-of (f1 f2) or opt f
- **Remove duplicated feature expression is not always a safe operation**
  For example, let fe be all (one-of(f1 f2) one-of(f1 f2)); with the system just presented, fereduces to one-of(f1 f2). But we expect instead that fe reduces to one-of (f1 f2 (all f1 f2)).
- **Hard to implement**
  Implementing an algorithm that automatically manipulate features expressions is not a trivial task at all.

3 Our Approach: Algebraic Model

We think that it is better to define a language equivalent to feature diagrams in the sense that both of them have the same meaning. We know that a Feature Diagram (FD) denotes all possible configurations of a system [2], so, what we have to define is a textual language that denotes combinations (see denotational semantic in [4]). We have found that a known algebraic structure can be used to describe features. Any language intended to be used to describe features must have certain operations. We think that such a language must be equipped with at least two operations: one should give the possibility of choice between two descriptions, and the other should join two descriptions. Other operations could be added but these must always be included. We define an algebra that includes these basic operations (choice and join), and show that this algebra captures the notion of describing features.

Definition 3.0.5 (Algebraic Model)
Let F be a nonempty set, then we define $R_M$as follows: $R_M = (\mathcal{P}^2(F), \cup, \otimes, \emptyset, \{\emptyset\})$, with $\mathcal{P}(F)$ being the powerset of F, $\mathcal{P}^2(F)$, the powerset of $\mathcal{P}(F)$, the usual set union $\cup$, the empty set $\emptyset$ and $\otimes$ defined as follows:

$$D_1 \otimes D_2 = \{d_1 \cup d_2 \mid d_1 \in D_1 \wedge d_2 \in D_2\}$$

Proposition 3.0.1
R_M is a commutative semi-ring.

Definition 3.0.6 (D^*)
Let D be a set,

$$D^* = \sum_{i \geq 1} D^i$$

where

$$\begin{array}{lcl} D^0 & = & \{\emptyset\} \\ D^{n+1} & = & D^n \otimes D \end{array}$$

We could describe features operating directly with this algebra, but we think that is better to hide it and work with expressions in a syntactic way. In order to do that, we define the language of feature expressions and then provide a simple way to manipulate them.

Definition 3.0.7 (Feature Expressions)
If F is a set, and we consider each element of F a feature, then the set fexp of feature expressions over F is defined as follows.

$$\begin{array}{llll} & \hat{\emptyset} & & \in \mathsf{fexp} \\ & \lambda & & \in \mathsf{fexp} \\ \text{if} & f \in F & \text{then} & f \in \mathsf{fexp} \\ \text{if} & f_{e1}, f_{e2} \in \mathsf{fexp} & \text{then} & (f_{e1} + f_{e2}) \in \mathsf{fexp} \\ \text{if} & f_{e1}, f_{e2} \in \mathsf{fexp} & \text{then} & (f_{e1}\ f_{e2}) \in \mathsf{fexp} \end{array}$$

The semantics of fexp is given using $\mathrm{R_M}$

.

Definition 3.0.8 (Interpretation)
A feature expression denotes a set of configurations.

$$\begin{array}{lcl} [\![]\!]_{\mathrm{F}} : \mathsf{fexp} & \rightarrow & \mathcal{P}^2(\mathrm{F}) \\ [\![\hat{\emptyset}]\!]_{\mathrm{F}} & = & \emptyset \\ [\![\lambda]\!]_{\mathrm{F}} & = & \{\emptyset\} \\ [\![\mathrm{f}]\!]_{\mathrm{F}} & = & \{\{\mathrm{f}\}\} \\ [\![\mathrm{f_{e1}} + \mathrm{f_{e2}}]\!]_{\mathrm{F}} & = & [\![\mathrm{f_{e1}}]\!]_{\mathrm{F}} \cup [\![\mathrm{f_{e2}}]\!]_{\mathrm{F}} \\ [\![\mathrm{f_{e1}f_{e2}}]\!]_{\mathrm{F}} & = & [\![\mathrm{f_{e1}}]\!]_{\mathrm{F}} \otimes [\![\mathrm{f_{e2}}]\!]_{\mathrm{F}} \end{array}$$

Note 3.0.1 (Regular Expressions) The definition of feature expressions is almost indenticaly to the definition of regular expressions [1].

A feature expression is denoting a set of possible configurations and sometimes it is desiarable to make it explicit.

Definition 3.0.9 (Disjunctive Normal Form)
A feature expression fe is in Disjunctive Normal Form if it has the following form:

$$\mathsf{fe} = \sum_{i=1}^{n} \mathrm{f_i}, \text{where } \mathrm{f_i} \in (\mathrm{F} \cup \{\emptyset, \lambda\})$$

We also say that fe is an n-term expression.

$$\sum_{i=1}^{n} \mathrm{f}_i = \begin{cases} \hat{\emptyset} & \text{if } n < 1 \\ \mathrm{f}_1 + \ldots + \mathrm{f}_n & \text{if } n \geq 1 \end{cases}$$

We will say that two expressions are equivalent (and note $\mathrm{f_{e1}} \equiv \mathrm{f_{e2}}$) if and only if their meanings are the same.

Definition 3.0.10 (Feature Expressions equivalence)
Let $\mathrm{f_{e1}}$, $\mathrm{f_{e2}} \in \mathsf{fexp}$, then

$$\mathrm{f_{e1}} \equiv \mathrm{f_{e2}} \text{ iff } [\![\mathrm{f_{e1}}]\!]_{\mathrm{F}} = [\![\mathrm{f_{e2}}]\!]_{\mathrm{F}}$$

Sometimes it is useful to combine features of a given expression.

Definition 3.0.11 (fe^*)
Let fe be a feature expression,

$$\mathsf{fe}^* = \sum_{i \geq 1} \mathsf{f}^i$$

where

$$\begin{array}{lcl} \mathsf{fe}^0 & = & \lambda \\ \mathsf{fe}^{n+1} & = & \mathsf{fe}^n \mathsf{fe} \end{array}$$

How many times do we have to combine a given feature expression with it self to get all possible combinations between its atomic features? The definition of n-regularity captures this.

Definition 3.0.12 (n-regularity)
A feature expression fe is n-regular iff $\mathsf{fe}^n = \mathsf{fe}^{n+1}$

Proposition 3.0.2 (Properties)
We will make use of several interesting properties:

1. $f_{e1}\ (f_{e2}\ f_{e3}) \equiv (f_{e1}\ f_{e2})\ f_{e3}$
2. $f_e\lambda \equiv \lambda f_e \equiv f_e$
3. $f_{e1}\ f_{e2} \equiv f_{e2}\ f_{e1}$
4. $f_{e1} + f_{e2} \equiv f_{e2} + f_{e1}$
5. $f_{e1} + (f_{e2} + f_{e3}) \equiv (f_{e1} + f_{e2}) + f_{e3}$
6. $f_e + \hat{\emptyset} \equiv \hat{\emptyset} + f_e \equiv f_e$
7. $f_e\ \hat{\emptyset} \equiv \hat{\emptyset}\ f_e \equiv \hat{\emptyset}$
8. $f_{e1}\ (f_{e2} + f_{e3}) \equiv f_{e1}\ f_{e2} + f_{e1}\ f_{e3}$
9. $f_e + f_e \equiv f_e$
10. $(f_{e1} + f_{e2})^n \equiv \sum_{i=0}^{n} f_{e1}^{n-i} f_{e2}^{i}$
11. if f_e n-term expression then f_e is n-regular

The first eight rules come directly from the semi-ring nature of the semantic structure. Rule nine establish that two options with the same meaning can be reduced to one. Rule ten is a consequence of the above rules. The last one says that a feature expression in disjunctive normal form contains all the combinations between its atomic features.

If we extend the way of constructing features expressions with a $\star$ operation on the following way

$$\text{if } f_e \in \mathsf{fexp} \text{ then } f_e^* \in \mathsf{fexp}$$

with its interpretation as $[\![f_e^*]\!]_F = [\![f_e]\!]_F^*$, then we also have the following properties:

Proposition 3.0.3

1. if f_e n-regular then $f_e^* \equiv \sum_{i=0}^{n} f_e^i$
2. $\hat{\emptyset}^* \equiv \lambda$
3. $f_e^* + f_e \equiv f_e^*$
4. $(f_{e1} + f_{e2})^* \equiv (f_{e1}^* f_{e2}^*)^*$
5. $f_e^{**} \equiv f_e^*$

Proposition 3.0.4 (Compositional Principle)

This property says that we can replace any feature expression with an equivalent one, and the resulting expression will be aquivalent to the original.

$$\text{if } f_e \equiv f_{e'} \text{ then } \begin{cases} f_e^* & \equiv & f_{e'}^*, \\ f_{e1} + f_e & \equiv & f_{e1} + f_{e'}, \\ f_{e1} f_e & \equiv & f_{e1} f_{e'}, \end{cases}$$

This property can also be stated saying that the equivalence relation between feature expressions is a congruence.

Definition 3.0.13 (Variability)

$$\pi(f_e) = |[\![f_e^*]\!]_F|$$

e.g, the variability of a feature expression is the number of posible configurations that it represents. For example, the variability for the feature expression for Car expression is 36 (see figure **??**).

Proposition 3.0.5

The following relation holds,

$$f_e^{\pi(f_e)} = f_e^{\pi(f_e)+1}$$

This states that f_e is a $\pi(f_e)$-term expression (the variability of a feature expression coincides with the number of possible configurations between the atomic features that compose the expression)

3.1 Constraints

The purpose of constraints is to further limit the variability of a system. As defined in [5], a constraint can have one of the following forms:

- f_1 **requires** f_2: if feature f_1 is present, then feature f_2 must be present as well.
- f_1 **excludes** f_2: if feature f_1 is present, then feature f_2 must not be present.
- **include** f: feature f must be present.
- **exclude** f: feature f must not be present.

The first two kinds of constraints are called diagram constraints since they express fixed, inherent dependencies between features in a diagram. The last two kinds of constraints are called user constraints since they express the user requirements regarding presence or absence of a feature. The user constraints may vary between subsequent uses of the feature diagrams [5]. This leads to the notion of satisfaction that determines for each feature expression whether it satisfies a constraint.

Definition 3.0.14 (Satisfability)
Let $\zeta(\mathsf{fe}, c)$ means that fe satisfies a constraint c. We can define $\zeta(\mathsf{fe}, c)$ as follows:

$$\begin{array}{lcl} \zeta(\mathsf{fe}, \mathrm{f}_1 \textbf{ requires } \mathrm{f}_2) & = & \text{if } \mathrm{f}_1 \in [\![\mathsf{fe}]\!]_{\mathrm{F}} \text{ then } \mathrm{f}_2 \in [\![\mathsf{fe}]\!]_{\mathrm{F}} \\ \zeta(\mathsf{fe}, \mathrm{f}_1 \textbf{ excludes } \mathrm{f}_2) & = & \text{if } \mathrm{f}_1 \in [\![\mathsf{fe}]\!]_{\mathrm{F}} \text{ then } \mathrm{f}_2 \notin [\![\mathsf{fe}]\!]_{\mathrm{F}} \\ \zeta(\mathsf{fe}, \textbf{include } \mathsf{f}) & = & \mathsf{f} \in [\![\mathsf{fe}]\!]_{\mathrm{F}} \\ \zeta(\mathsf{fe}, \textbf{exclude } \mathsf{f}) & = & \mathsf{f} \notin [\![\mathsf{fe}]\!]_{\mathrm{F}} \end{array}$$

We could extend ζ to work over sets of constraints in the following way:

$$\zeta(\mathsf{fe}, C) = \forall c \in C : \zeta(\mathsf{fe}, c)$$

If we introduce the following two constraints in the car example: pullsTrailer requires highPower and include pullsTrailer, the original 36 possibilities will be reduced to just 6 (see figure 2).

3.2 FDL*

We are now in position to define FDL* by reusing FDL's syntax. The language FDL* hides the algebra and makes it more readable for programmers. The main advantage of this definition is that all the properties of FDL* will follow directly from the properties of the underlying model.

Definition 3.0.15 (FDL*)

$$\begin{array}{lllll} & \mathsf{Null} & & & \in \mathsf{fexp} \\ & \mathrm{f} \in \mathrm{F} & \text{then } \mathrm{f} & & \in \mathsf{fexp} \\ \text{if} & \mathrm{f_e} \in \mathsf{fexp} & \text{then } \mathrm{f_e}? & & \in \mathsf{fexp} \\ \text{if} & \mathrm{f_{e1}}, \mathrm{f_{e2}} \in \mathsf{fexp} & \text{then } \mathsf{all}\ \mathrm{f_{e1}} \mathrm{f_{e2}} & & \in \mathsf{fexp} \\ \text{if} & \mathrm{f_{e1}}, \mathrm{f_{e2}} \in \mathsf{fexp} & \text{then } \mathsf{oneOf}\ \mathrm{f_{e1}} \mathrm{f_{e2}} & & \in \mathsf{fexp} \\ \text{if} & \mathrm{f_{e1}}, \mathrm{f_{e2}} \in \mathsf{fexp} & \text{then } \mathsf{moreOf}\ \mathrm{f_{e1}} \mathrm{f_{e2}} & & \in \mathsf{fexp} \end{array}$$

Observe that in contrast to FDL there is a nullary operation (Null), this operation make operating with feature expression easier; all and oneOf are binary operations; they can be generalized to n-ary ones as in definition 3.0.17.

The semantics of language FDL* is given in the next definition.

Definition 3.0.16 (Interpretation)

$$
\begin{array}{lcl}
[\![]\!]_{\mathrm{fdl}} : \mathsf{fexp} \rightarrow \mathcal{P}^2(\mathrm{F})) & & \\
[\![\mathsf{Null}]\!]_{\mathrm{fdl}} & = & [\![\lambda]\!]_{\mathrm{F}} \\
[\![\mathrm{f}]\!]_{\mathrm{fdl}} & = & [\![\mathrm{f}]\!]_{\mathrm{F}} \\
[\![\mathrm{f_e}?]\!]_{\mathrm{fdl}} & = & [\![\lambda + \mathrm{f_e}]\!]_{\mathrm{F}} \\
[\![\mathsf{all}\ \mathrm{f_{e1}f_{e2}}]\!]_{\mathrm{fdl}} & = & [\![\mathrm{f_{e1}f_{e2}}]\!]_{\mathrm{F}} \\
[\![\mathsf{oneOf}\ \mathrm{f_{e1}f_{e2}}]\!]_{\mathrm{fdl}} & = & [\![\mathrm{f_{e1}} + \mathrm{f_{e2}}]\!]_{\mathrm{F}} \\
[\![\mathsf{moreOf}\ \mathrm{f_{e1}f_{e2}}]\!]_{\mathrm{fdl}} & = & [\![\mathrm{f_{e1}} + \mathrm{f_{e1}f_{e2}} + \mathrm{f_{e2}}]\!]_{\mathrm{F}}
\end{array}
$$

We could also calculate a version of $[\![]\!]_{fdl}$ in an explicit form, resulting in:

$$
\begin{array}{lcl}
[\![]\!]_{\mathrm{fdl}} : \mathsf{fexp} \rightarrow \mathcal{P}^2(\mathrm{F})) & & \\
[\![\mathsf{Null}]\!]_{\mathrm{fdl}} & = & \{\emptyset\} \\
[\![\mathrm{f}]\!]_{\mathrm{fdl}} & = & \{\{\mathrm{f}\}\} \\
[\![\mathrm{f_e}?]\!]_{\mathrm{fdl}} & = & [\![\mathsf{oneOf}\ \mathsf{Null}\ \mathrm{f_e}]\!]_{\mathrm{fdl}} \\
[\![\mathsf{all}\ \mathrm{f_{e1}f_{e2}}]\!]_{\mathrm{fdl}} & = & [\![\mathrm{f_{e1}}]\!]_{\mathrm{fdl}} \otimes [\![\mathrm{f_{e2}}]\!]_{\mathrm{fdl}} \\
[\![\mathsf{oneOf}\ \mathrm{f_{e1}f_{e2}}]\!]_{\mathrm{fdl}} & = & [\![\mathrm{f_{e1}}]\!]_{\mathrm{fdl}} \cup [\![\mathrm{f_{e2}}]\!]_{\mathrm{fdl}} \\
[\![\mathsf{moreOf}\ \mathrm{f_{e1}f_{e2}}]\!]_{\mathrm{fdl}} & = & [\![\mathsf{oneOf}\ (\mathrm{f_{e1}}(\mathsf{oneOf}\ (\mathsf{all}\ \mathrm{f_{e1}f_{e2}})\mathrm{f_{e2}})]\!]_{\mathrm{F}}
\end{array}
$$

Corollary 3.0.1

- $\mathrm{f_e}? \equiv \lambda + \mathrm{f_e}$
- $\mathsf{all}\ \ \mathrm{f_e}1\ \mathrm{f_{e2}} \equiv \mathrm{f_{e1}}\ \mathrm{f_{e2}}$
- $\mathsf{oneOf}\ \ \mathrm{f_e}1\ \ \mathrm{f_{e2}} \equiv \mathrm{f_{e1}} + \mathrm{f_{e2}}$
- $\mathsf{moreOf}\ \ \mathrm{f_e}1\ \mathrm{f_{e2}} \equiv \mathrm{f_{e1}} + \mathrm{f_{e1}f_{e2}} + \mathrm{f_{e2}}$

Proposition 3.0.6 (Derivation of properties)
The following properties are valid in FDL*

- $\mathsf{all}\ \ \mathrm{f_{e1}}\ (\mathrm{f_{e2}}?) \equiv \mathsf{oneOf}\ \ \mathrm{f_{e1}}\ (\mathsf{all}\ \ \mathrm{f_{e1}}\ \mathrm{f_{e2}})$
- $\mathsf{all}\ \ \mathrm{f_{e1}}\ (\mathsf{oneOf}\ \ \mathrm{f_{e2}}\ \mathrm{f_{e3}}\ )$
  $\equiv \mathsf{oneOf}\ \ (\mathsf{all}\ \ \mathrm{f_{e1}}\ \mathrm{f_{e2}})\ (\mathsf{all}\ \ \mathrm{f_{e1}}\ \mathrm{f_{e3}})$
- $\mathsf{all}\ \ \mathrm{f_{e1}}\ (\mathsf{moreOf}\ \ \mathrm{f_{e2}}\ \mathrm{f_{e3}})$
  $\equiv \mathsf{oneOf}\ \ (\mathsf{all}\ \ \mathrm{f_{e1}}\ \mathrm{f_{e2}})\ (\mathsf{oneOf}\ \ (\mathsf{all}\ \ \mathrm{f_{e1}}\ \mathrm{f_{e3}})\ (\mathsf{all}\ \ \mathrm{f_{e1}}\ (\mathsf{all}\ \ \mathrm{f_{e2}}\ \mathrm{f_{e3}}))$

We could define operations abbreviating common combinations, giving us the usefullness of simplicity, but not adding extra expressive power.

Definition 3.0.17

$$
\begin{array}{lcl}
\mathsf{All}_{i=1}^{n}\mathrm{f_{ei}} & = & \begin{cases} \mathrm{Null} & \text{if } n < 1 \\ \mathsf{all}\ \mathrm{f_{e1}}(\mathsf{all}\ \mathrm{f_{e2}} \ldots \mathrm{f_{en}}) & \text{if } n \geq 1 \end{cases} \\
\mathsf{OneOf}_{i=1}^{n}\mathrm{f_{ei}} & = & \begin{cases} \mathrm{Null} & \text{if } n < 1 \\ \mathsf{oneOf}\ \mathrm{f_{e1}}(\mathsf{oneOf}\ \mathrm{f_{e2}} \ldots \mathrm{f_{en}}) & \text{if } n \geq 1 \end{cases} \\
\mathsf{MoreOf}_{i=1}^{n}\mathrm{f_{ei}} & = & \begin{cases} \mathrm{Null} & \text{if } n < 1 \\ \mathsf{moreOf}\ \mathrm{f_{e1}}(\mathsf{moreOf}\ \mathrm{f_{e2}} \ldots \mathrm{f_{en}}) & \text{if } n \geq 1 \end{cases}
\end{array}
$$

A feature expression is denoting a set of possible configurations and sometimes it is desirable to make it explicit.

Definition 3.0.18 (Disjunctive Normal Form)
A feature expression fe is in Disjunctive Normal Form if it has the following form:

$$\mathsf{OneOf}_{i=1}^{n}(\mathsf{All}_{j=1}^{m}\mathrm{f_j}), \mathrm{with f_j in\ F}.$$

We also say that fe is an n-term expression.

An expression in disjunctive normal form may have repeated features. For that reason, we define the notion of normalization, that allow to remove those uneeded duplications.

Definition 3.0.19 (Normalization)
A feature expression fe is normalized if it has the following form:

$$\mathsf{OneOf}_{i=1}^{n} \mathrm{f_{ei}}$$

where each $\mathrm{f_{ei}}$ has the form

$$\mathsf{All}_{j=1}^{m} \mathrm{f_j}, \text{with } \mathrm{f_j} \text{ in F and } \mathrm{f_j} \neq \mathrm{f_k}, \text{for all} j \neq k$$

and $\mathrm{f_{ei}} \neq \mathrm{f_{ej}}$, for all i ≠ j. We also say that fe is an n-term expression.

It is an easy task to prove that the normalization and expansion rules given in Section 1 are properties over our model, proving in that way that FDL* is at least as powerful as FDL.

4 Specification and Parsings

We have abused with some notations, for example: the Car example was not writing in any of the languages we defined, in the example we make use of the metavariables: Car, Transmission, Engine, HorsePower. To eliminate them, just replace all ocurrency of identifiers by the content of the referenced metavariable. We can write a parser that suport this feature rather than one just for feature expressions, allowing to write FDL* programs using metavariables.

5 Implementation

Once we have defined the language of fexpof feature expressions, it is now easy to implement a function to reduce a feature expression to its disjunctive normal form and a function that evaluates a feature expression in almost any programming language; we have chosen Haskell [3], a functional programming language, because of its simplicity. We divide the program in two modules: the main module is FeatureExpression, implementing FeatureExpression data types and two operations: eval (which evals a feature expression to its denotational value) and red2DNF (which reduce a feature expression to its disjunctive normal form). The module Set implements the Set Abstract Data Type; we use this module because the meaning of a feature expression is a set. Now, it is possible to obtain an implementation for the language FDL*by deriving the fexpone. We present the code in Figures 3 to 6.

6 Examples

We take the same example as [5] in order to provide an easy bridge for comparison of both approaches.

```
Car          = All {carBody, Trasmission, Engine, HorsePower, pullsTrailer?}
Transmission = oneOf automatic manual
Engine       = moreOf electric gasoline
HorsePower   = OneOf {lowPower, mediumPower, highPower}
```

From this we obtain that the set of features is:

```
F = { carBody, automatic, manual, electric, gasoline,lowPower,
      mediumPower,highPower,pullsTrailer}
```

As the real importance of a feature expression resides on its structure, we can replace each of the features in the example by a single letter, simplifying in this way all the equations and calculations.

a = carBody
b = automatic
c = manual
d = electric
e = gasoline
f = lowPower
g = mediumPower
h = highPower
i = pullsTrailer

Note 6.0.2 Car ≡ a (b + c) (d + e + d e) (f + g + h) (i + j)

To obtain a feature expression in disjunctive normal form just repeatedly apply property 8, and to remove duplications apply properties 3 and 4.

The disjuctive normal form of expression Car is obtained as:

$$abdfi + abdfj + abdgi + abdgj +$$
$$abdhi + abdhj + abefi + abefj +$$
$$abegi + abegj + abehi + abehj +$$
$$abdefi + abdefj + abdegi +$$
$$abdegj + abdehi + abdehj +$$
$$acdfi + acdfj + acdgi + acdgj +$$
$$acdhi + acdhj + acefi + acefj +$$
$$acegi + acegj + acehi + acehj +$$
$$acdefi + acdefj + acdegi +$$
$$acdegj + acdehi + acdehj$$

The full feature expression is given in figure 1.

7 Conclusions

The semantic of languages for describing features is captured by an algebraic structure (R_M) and the language of expressions over this structure is the minimal complete of them. It is possible to define new languages with more operators in terms of the basic ones. This has two mayor benefits: the properties of the model are inherited and the implementation of them can be derived from the basic one. We have formalized the class of languages for describing features and defined a new language (FDL*) with the idea of formalize, ((correct)) and simplify a language already defined in the literature.

References

[1] John E. Hopcroft Jeffrey D. Ullman. *Formal Languages and their relation to Automata*. Addison-Wesley Publishing Company, 1969.

[2] K. C. Kang S. G. Cohen J. A. Hess W. E. Novak A. S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical report, Software Engineering Institute, Carnegie Mellon University, 1990.

[3] Simon Peyton Jones and John Hughes (editors). Haskell 98: A non-strict, purely functional language, February 1999.
URL: `http://www.haskell.org/onlinereport/`.

[4] John C. Reynolds. *Theories of Programming Languages*. Cambridge University Press, 1998.

[5] Arie van Duersen Paul Klint. Domain-specific language design requires feature descriptions. *ACM Computing Surveys*, 2000.

```
oneof (
  all(carBody automatic electric                lowPower    pullsTrailer)
  all(carBody automatic electric                lowPower)
  all(carBody automatic electric                mediumPower pullsTrailer)
  all(carBody automatic electric                mediumPower)
  all(carBody automatic electric                highPower   pullsTrailer)
  all(carBody automatic electric                highPower)
  all(carBody automatic gasoline                lowPower    pullsTrailer)
  all(carBody automatic gasoline                lowPower)
  all(carBody automatic gasoline                mediumPower pullsTrailer)
  all(carBody automatic gasoline                mediumPower)
  all(carBody automatic gasoline                highPower   pullsTrailer)
  all(carBody automatic gasoline                highPower)
  all(carBody automatic electric  gasoline      lowPower    pullsTrailer)
  all(carBody automatic electric  gasoline      lowPower)
  all(carBody automatic electric  gasoline      mediumPower pullsTrailer)
  all(carBody automatic electric  gasoline      mediumPower)
  all(carBody automatic electric  gasoline      highPower   pullsTrailer)
  all(carBody automatic electric  gasoline      highPower)
  all(carBody manual    electric                lowPower    pullsTrailer)
  all(carBody manual    electric                lowPower)
  all(carBody manual    electric                mediumPower pullsTrailer)
  all(carBody manual    electric                mediumPower)
  all(carBody manual    electric                highPower   pullsTrailer)
  all(carBody manual    electric                highPower)
  all(carBody manual    gasoline                lowPower    pullsTrailer)
  all(carBody manual    gasoline                lowPower)
  all(carBody manual    gasoline  mediumPower   pullsTrailer)
  all(carBody manual    gasoline                mediumPower)
  all(carBody manual    gasoline                highPower   pullsTrailer)
  all(carBody manual    gasoline                highPower)
  all(carBody manual    electric  gasoline      lowPower    pullsTrailer)
  all(carBody manual    electric  gasoline      lowPower)
  all(carBody manual    electric  gasoline      mediumPower pullsTrailer)
  all(carBody manual    electric  gasoline      mediumPower)
  all(carBody manual    electric  gasoline      highPower   pullsTrailer)
  all(carBody manual    electric  gasoline      highPower)
)
```

Figure 1: Disjunctive normal form for Car (36 disjuncts)

```
oneof (
  all(carBody automatic electric                highPower pullsTrailer)
  all(carBody automatic gasoline                highPower pullsTrailer)
  all(carBody automatic electric  gasoline      highPower pullsTrailer)
  all(carBody manual    electric                highPower pullsTrailer)
  all(carBody manual    gasoline                highPower pullsTrailer)
  all(carBody manual    electric  gasoline      highPower pullsTrailer)
)
```

Figure 2: Reduced feature expression for Car (6 disjuncts)

A Appendix

A.1 Proofs of propositions

Proposition 3.0.1 $\mathbf{R_M}$ is a commutative semi-ring. We can separate the proof in several smaller ones; some of those will follow directly from well known properties of the union operation. We give proofs of the nontrivial ones.

- $D_1 \otimes (D_2 \otimes D_3) = (D_1 \otimes D_2) \otimes D_3$

$$
\begin{array}{ll}
D_1 \otimes (D_2 \otimes D_3) & = \\
D_1 \otimes \{d_2 \cup d_3 | d_2 \in D_2 \wedge d_3 \in D_3\} & = \\
\{d_1 \cup d_{23} | d_1 \in D_1 \wedge d_{23} \in \{d_2 \cup d_3 | d_2 \in D_2 \wedge d_3 \in D_3\}\} & = \\
\{d_1 \cup (d_2 \cup d_3) | d_1 \in D_1 \wedge (d_2 \in D_2 \wedge d_3 \in D_3)\} & = \\
\{(d_1 \cup d_2) \cup d_3) | (d_1 \in D_1 \wedge d_2 \in D_2) \wedge d_3 \in D_3\} & = \\
\{d_{12} \cup d_3 | d_{12} \in \{d_1 \cup d_3 | d_1 \in D_1 \wedge d_2 \in D_2\} \wedge d_3 \in D_3\} & = \\
\{d_1 \cup d_2 | d_1 \in D_1 \wedge d_2 \in D_2\} \otimes D_3 & = \\
(D_1 \otimes D_2) \otimes D_3 &
\end{array}
$$

To complete the proof, the following properties must hold. The proofs can be easily checked.

- $D_1 \cup (D_2 \otimes D_3) = (D_2 \otimes D_3) \cup D_1 = (D_1 \otimes D_2) \cup (D_1 \otimes D_3)$
- $D \otimes \{\emptyset\} = \{\emptyset\} \otimes D$
- $D_1 \otimes D_2 = D_2 \otimes D_1$
- $D_1 \cup (D_2 \cup D_3) = (D_1 \cup D_2) \cup D_3$
- $D \cup \emptyset = \emptyset \cup D$
- $D_1 \cup D_2 = D_2 \cup D_1$
- $D \otimes \emptyset = \emptyset \otimes D$

Lemma A.0.1 Let D be a set, if $\mathrm{D}^n = \mathrm{D}^{n+1}$ then $\mathrm{D}^n = \mathrm{D}^{n+k}$, for all $\mathrm{k} \geq 1$.

Proposition ?? $\mathrm{D}^{|\mathrm{D}|} = \mathrm{D}^{|\mathrm{D}+1|}$, $\forall \mathrm{D} \neq \emptyset$
Proof by Induction over |D|
Base Case: |D|= 1, (i.e. D = $\{d\}$)

$$\mathrm{D}^1 = \{d\} = \{d \cup d\} = \{d\} \otimes \{d\} = \mathrm{D}^1 \otimes \mathrm{D}^1 = \mathrm{D}^2$$

Inductive Case:
$|\mathrm{D}| = k + 1$, (i.e. $\mathrm{D} = \{d_1, \ldots, d_k, d_{k+1}\}$)

IH) $\mathrm{D}^{|\mathrm{D}|} = \mathrm{D}^{|\mathrm{D}+1|}, \forall\ \mathrm{D} \in \mathcal{P}^2(\mathrm{F}) - \emptyset$, with $|\mathrm{D}| = k$

IT) $\mathrm{D}^{|\mathrm{D}|} = \mathrm{D}^{|\mathrm{D}+1|}, \forall\ \mathrm{D} \in \mathcal{P}^2(\mathrm{F}) - \emptyset$, with $|\mathrm{D}| = k + 1$

$$
\begin{array}{lll}
D^{k+1} & = & \\
\{d_1, \ldots, d_k, d_{k+1}\}^{k+1} & = & \\
(\{d_1, \ldots, d_k\} \cup \{d_{k+1}\})^{k+1} & = & \\
\sum_{i=0}^{k+1} \{d_1, \ldots, d_k\}^{k+1-i} \{d_{k+1}\}^i & = & \\
\sum_{i=1}^{k} \{d_1, \ldots, d_k\}^{k+1-i} \{d_{k+1}\}^i \cup \{d_1, \ldots, d_k\}^{k+1} \cup \{d_{k+1}\}^{k+1} & & \\
 & = & \text{(by HI and Lemma 1)} \\
\sum_{i=1}^{k} \{d_1, \ldots, d_k\}^{k+1-i} \{d_{k+1}\}^i \cup \{d_1, \ldots, d_k\}^{k} \cup \{d_{k+1}\}^{k} & = & \\
\sum_{i=0}^{k} \{d_1, \ldots, d_k\}^{k-i} \{d_k\}^i & = & \\
(\{d_1, \ldots, d_k\} \cup \{d_{k+1}\})^{k} & = & \\
\{d_1, \ldots, d_k, d_{k+1}\}^{k} & = & \\
D^k & &
\end{array}
$$

A.2 Implementation

```
module Set (Set,emptySet,isEmpty,add,union,belong, product, singleton) where
  data Eq a => Set a = Set [a]

  emptySet:: Eq a => Set a
  emptySet = Set []

  isEmpty (Set [])     = True
  isEmpty (Set (x:xs)) = False

  belongs x (Set xs)   = x `elem` xs
  add     x (Set xs)   = if belongs x (Set xs)
                         then Set xs
                         else Set (x:xs)
  union  (Set xs) (Set ys) = addAll xs (Set ys)
      where addAll [] s = s
            addAll (x:xs) s = addAll xs (add x s)
  product f (Set xs) (Set ys) = Set [f x y | x <- xs, y <- ys]
  singleton x = add x emptySet
```

Figure 3: Haskell implementation (module for sets)

```
module FeatureExpression(FExp,Feature,eval,red2DNF) where
  import Set

  type Feature = String
  data FExp = EmptySet
            | Lambda
            | Atomic Feature
            | Plus FExp FExp
            | Prod FExp FExp

  eval:: FExp -> Set (Set Feature)
  eval EmptySet        = emptySet
  eval Lambda          = singleton emptySet
  eval (Atomic fe)     = singleton (singleton fe)
  eval (Plus fe1 fe2)  = (eval fe1) `union` (eval fe2)
  eval (Prod   fe1 fe2)= sproduct union (eval fe1) (eval fe2)

  red2DNF:: FExp -> FExp
  red2DNF EmptySet       = EmptySet
  red2DNF Lambda         = Lambda
  red2DNF (Atomic f)     = Atomic  f
  red2DNF (Plus fe1 fe2) = Plus (red2DNF fe1) (red2DNF fe2)
  red2DNF (Prod fe1 fe2) = distribute (Prod (red2DNF fe1) (red2DNF fe2))

  distribute (Prod (Plus fa fb) fc) =
       Plus (distribute (Prod fa fc))(distribute (Prod fb fc))
  distribute (Prod fa (Plus fb fc)) =
       Plus (distribute (Prod fa fb))(distribute (Prod fa fc))
  distribute fe = fe
```

Figure 4: Haskell implementation (module for features)

```
module FDLExp (FDLExp,red2DNF,eval,normalize) where
  import Set

  type Feature = String

  data FDLExp = Null
              | Atomic Feature
              | Opt    FDLExp
              | OneOf  FDLExp FDLExp
              | All    FDLExp FDLExp
              | MoreOf FDLExp FDLExp   deriving Eq

  eval:: FDLExp -> Set (Set Feature)
  eval Null             = singleton emptySet
  eval (Atomic fe)      = singleton (singleton fe)
  eval (OneOf fe1 fe2)  = (eval fe1) `union` (eval fe2)
  eval (All   fe1 fe2)  = product union (eval fe1) (eval fe2)
  eval (Opt    fe)      = (eval Null) `union` (eval fe)
  eval (MoreOf fe1 fe2) = let ev1 = eval fe1
                              ev2 = eval fe2
                          in ev1 `union` (product ev1 ev2) `union` ev2

  red2DNF:: FDLExp -> FDLExp
  red2DNF Null            = Null
  red2DNF (Atomic f)      = Atomic  f
  red2DNF (Opt fe)        = OneOf  Null fe
  red2DNF (OneOf fe1 fe2) = OneOf (red2DNF fe1) (red2DNF fe2)
  red2DNF (MoreOf fe1 fe2)= OneOf (red2DNF fe1)
                                  (OneOf (red2DNF (All (red2DNF fe1) (red2DNF fe2)))
                                         (red2DNF fe2))
  red2DNF (All fe1 fe2)   = distribute (All (red2DNF fe1) (red2DNF fe2))

  distribute (All (OneOf fa fb) fc) = OneOf (distribute fa fc)(distribute fb fc)
  distribute (All fa (OneOf fb fc)) = OneOf (distribute fa fb)(distribute fa fc)
  distribute fe                     = fe
```

Figure 5: Haskell implementation (module for FDL* expressions, part 1)

```
normalize:: FExp -> FExp
normalize Null          = Null
normalize (Atomic f)    = Atomic f
normalize (All f1 f2)   = allsFromList
                           (remDupsInList (allsToList f1 ++
                                           allsToList f2))
normalize (OneOf f1 f2) = let f1'= normalize f1
                              f2'= normalize f2
                           in oneOfFromList
                               (remDupsInList (oneOfsToList f1' ++
                                               oneOfsToList f2'))
allsToList Null        = []
allsToList (Atomic f)  = [Atomic f]
allsToList (All f1 f2) = (allsToList f1) ++ (allsToList f2)

oneOfsToList Null         = [Null]
oneOfsToList (Atomic f)   = [Atomic f]
oneOfsToList (All f1 f2)  = [All f1 f2]
oneOfsToList (OneOf f1 f2)= (oneOfsToList f1) ++ (oneOfsToList f2)

allsFromList    xs = treeFromList All xs

oneOfsFromList  xs = treeFromList OneOf xs

treeFromList n []      = Null
treeFromList n [f]     = f
treeFromList n (f1:fs) = n f1 (treeFromList n fs)

remDupsInList [] = []
remDupsInList (x:xs) = if x `elem` xs
                         then remDupsInList xs
                         else x: remDupsInList xs
```

Figure 6: Haskell implementation (module for FDL* expressions, part 2)

Using Stochastic NTCC to Model Biological Systems

Carlos Olarte and Camilo Rueda
Javeriana University, Dept. Computer Science.
Cali, Colombia
{caolarte,crueda}@atlas.puj.edu.co

Abstract

Concurrent process calculi are powerful formalisms for modeling concurrent systems. The mathematical style underlying process calculi allow to both model and verify properties of a system, thus providing a concrete design methodology for complex systems. `ntcc` , a constraints-based calculus for modeling temporal non-deterministic and asynchronous behavior of processes has been proposed recently. Process interactions in `ntcc` can be determined by partial information (i.e. constraints) accumulated in a global store. `ntcc` has also an associated temporal logic with a proof system that can be conveniently used to formally verify temporal properties of processes. We are interested in using `ntcc` to model the activity of genes in biological systems. In order to account for issues such as the basal rate of reactions or binding affinities of molecular components, we believe that stochastic features must be added to the calculus. In this paper we propose an extension of `ntcc` with various stochastic constructs. We describe the syntax and semantics of this extension together with the new temporal logic and proof system associated with it. We show the relevance of the added features by modeling a non trivial biological system: the gene expression mechanisms of the λ virus. We argue that this model is both more elaborate and compact than the stochastic π calculus model proposed recently for the same system.

Keywords: NTCC, Lambda-Switch, biological systems, concurrent process calculus, concurrent constraint programming

Resumen

Los cálculos de procesos concurrentes son un poderoso formalismo para modelar sistemas concurrentes. El soporte matemático del cálculo permite modelar y probar propiedades de un sistema, proveyendo así, una metodología concreta para el diseño de sistemas complejos. `ntcc` es un cálculo basado en restricciones propuesto recientemente para modelar comportamiento no determinístico y asíncrono. La interacción de los procesos en este cálculo es determinada por medio de información parcial (es decir restricciones) acumuladas en un almacén global de restricciones. Adicionalmente, `ntcc` cuenta con una lógica temporal y un sistema de inferencia que puede ser utilizado para verificar formalmente propiedades de los procesos. Nosotros estamos interesados en utilizar `ntcc` para modelar el comportamiento de los genes en los sistemas biológicos. Sin embargo, para ello se hace necesario la introducción de características estocásticas en el cálculo para modelar por ejemplo la afinidad entre las moléculas. En este artículo proponemos una extensión de `ntcc` con algunos operadores estocásticos. Para dicha extensión describimos la sintaxis , la semántica y una nueva lógica temporal junto con su sistema de inferencia. A partir de este, modelamos el mecanismo de expresión de los genes del virus λ y mostramos que dicho modelo es mas simple y compacto que el modelo propuesto recientemente utilizando el cálculo π estocástico.

Palabras claves: NTCC, Lambda-Switch, Sistemas biológicos, Cálculos de procesos concurrentes, Programación concurrente por restricciones

1 Introduction

We are interested in using soft computing techniques for modeling complex systems such as those arising frequently in biology. From a broad perspective we view soft computing as those techniques pertaining to systems that can best be described as a collection of processes dealing with partial information. What

"partial" means depends on the particular application. It can refer to being able to use partial knowledge of a state of affairs and to perform different kinds of guessing (bounded or unbounded non determinism, probabilistic choices, approximate answers). In this view, concurrency also belongs to this realm since it deals with partial information on the ordering of events. So does constraint programming which is based on the very idea of computing with *predicates* expressing different degrees of knowledge about variable values. Concurrent constraint (CC) process calculi [11] provides formal grounds to the integration of concurrency and constraints so that non trivial properties of concurrent systems can be expressed and proved. They are thus natural simulators to gain experience on different soft computing techniques.

We view biological phenomena at the molecular level as constructed from very complex interactions among a great number of concurrent processes acting at different biological scales.

Concurrent processes occurring in molecular biology exhibit a rich variety of synchronization schemes, calling into play different degrees of precision (i.e. partial information) about temporal or chemical relations involving them. The complexity of biological phenomena poses a great challenge to any computational formalism. We think that a suitable CC process calculus should provide a convenient framework to get insights into the right models to cope with this challenge.

We thus borrow concepts and techniques from concurrent processes modeling to define suitable computational calculi and analyze their behavior in real biological settings. What we gain from this *low level* approach is twofold. On the one hand, we are able to ground the development of simulation tools on a very precise formal foundation and by this means proposing coherent models of higher level biological structures and operations. On the other hand, our model can give us clues for constructing formal proofs of interesting properties of a given biological process.

We propose using a temporal non deterministic concurrent calculus (`ntcc` , see [6]) as a formal base to model timed gene activity processes in such a way that their biological properties can be formally proved. This goes in the same direction as the concurrent process calculi models of biological systems proposed recently ([9], [2], [3]).

The `ntcc` calculus inherits ideas from the tcc model [10], a formalism for reactive concurrent constraint programming. In tcc time is conceptually divided into *discrete intervals (or time-units).* In a particular time interval, a deterministic ccp process receives a stimulus (i.e. a constraint) from the environment, it executes with this stimulus as the initial store, and when it reaches its resting point, it responds to the environment with the resulting store. Also the resting point determines a residual process, which is then executed in the next time interval.

The `ntcc` calculus is obtained from tcc by adding *guarded-choice* for modeling non-determinism and an *unbounded but finite delay* operator for asynchrony. Computation in `ntcc` progresses as in tcc, except for the non-determinism and asynchrony induced by the new constructs. The calculus allows for the specification of temporal properties, and for modeling and expressing constraints upon the environment both of which are useful in proving properties of timed systems.

However, `ntcc` does not provide stochastic constructs. These are fundamental to faithfully model aspects such as the effect of reactions on concentration of particular components, affinities or distances. We thus propose orthogonal extensions of `ntcc` to account for the stochastic behavior of processes.

In this paper we are interested in showing how non trivial biological processes calling into action different forms of partial information can be modeled in `ntcc` extended with suitable stochastic constructs. We also investigate ways in which properties of a biological process can be formally proved. We are able to do this thanks to the logical nature of ntcc, which comes to the surface when we consider its relation with linear temporal logic: All the operators of ntcc correspond to temporal logic constructs. Since we extend `ntcc` , new linear temporal logic and proof system must also be provided. We propose both and use them to prove some properties of a gene regulation system called the lambda switch. Our model using the stochastic extension is both simpler and more complete than the one recently proposed in [3].

The main contributions of this paper are: 1) to define an orthogonal extension adding stochastic constructs to `ntcc` , 2) to couple the extended calculus with a suitable temporal logic and proof system, 3) to show how the expressiveness of the extended `ntcc` model allows faithful and simple descriptions of complex systems of interacting biological processes, such as the lambda switch and 4) showing that by modeling a gene activities system in the extended stochastic `ntcc` one inherits a well defined logical inference system (also proposed here) that can be used to prove interesting temporal properties (or lack thereof) of the system.

2 Background

2.1 NTCC Calculus

In concurrent constraint calculi such as ntcc, process interactions can be determined by partial information (i.e. constraints) accumulated in a global store. The particular type of constraints is not fixed but specified in a *constraint system* that is considered a parameter of the calculus.

2.1.1 Constraint System

A constraint represents a piece of partial information over a set of variables. For example, in constraint $x > 3$, the value of x is unknown but we can assert that it is greater than 3.
A constraint system provides a signature from which constraints can be constructed. It also provides an entailment relation ($\models$) over constraints where $c_1 \models c_2$ holds iff the information of c_2 can be inferred from c_1.
Formally, a constraint system is a tuple $\langle \sum, \Delta \rangle$ where $\sum$ is a signature (i.e a set of constants, functions and predicate symbols) and Δ is a consistent first-order theory over $\sum$ (i.e a set of sentences over $\sum$ having at least one model). Constraints can be viewed as first-order formulae over $\sum$ and $c \models d$ holds if the implication $c \Rightarrow d$ is valid in Δ [6]. For practical reasons the entailment relation must be decidable.
A constraint *store* is a a set of variables and a conjunction of formulae (i.e constrains) between them. It is used to share information between process and for synchronization purposes. The store is monotonically refined by adding information using *tell* operations of the calculus. For example, $tell(x < 2)$ adds constraint $x < 2$ to the store. Additionally, we can test if a constraint c can be entailed from the store by means of ask operations. For example, $ask(x < 5)$ tests whether $store \models x < 5$. The ask operation blocks when neither $store \models x < 5$ nor $store \models \neg(x < 5)$ holds.

2.1.2 Overview

`ntcc` [6] is a process calculus that extends tcc [10]. In both of them, processes share a common store of partial information [11]. Both ntcc and tcc have an explicit notion of (discrete) time. `ntcc` time is conceptually divided into *discrete intervals (or time-units).* In a particular time interval, a deterministic ccp process receives a stimulus (i.e. a constraint) from the environment, it executes with this stimulus as the initial store, and when it reaches its resting point, it responds to the environment with the resulting store. Also the resting point determines a residual process, which is then executed in the next time interval.

`ntcc` has been successfully used to model many real life system such as reactive system, robot behavior [5] and music composition [6].
Unlike tcc, `ntcc` includes constructs for modeling *nondeterminism* and *asynchrony.* A very important benefit of being able to specify non-deterministic and asynchronous behavior arises when modeling the interaction among several components running in parallel, in which one component is part of the environment of the others. This is frequent in biological settings. These systems often need non-determinism and asynchrony to be modeled faithfully.

2.1.3 Process Syntax

In this section we describe briefly the syntax of `ntcc` processes. See [6] for further details.
`ntcc` provides the following constructors:

- **tell** : adds new information to the constraints store. For example, the process $P_1 \stackrel{def}{=}$ **tell** $(c > 5)$ adds constraint $c > 5$ to the store.
- $\sum_{i \in 1..n}$ **when** c_i **do** P_i chooses non-deterministically a process P_i whose guard c_i is entailed by the store. For example, process $P_2 \stackrel{def}{=}$ **when** $(c < 3)$ **do tell** $(d = 5) +$ **when** $(e > 5)$ **do tell** $(d < 3)$ adds the information $d = 5$ when constraint $c < 3$ is entailed from the current store. On the other hand, if $e > 5$ is entailed, $d < 3$ is asserted. When both guards are entailed a non-deterministic choice is performed.

- Given two `ntcc` processes P and Q, process $P||Q$ represents the parallel composition between P and Q.
- **local** x **in** P behaves like P but the information of the variable x is local to P, i.e. P cannot see information about a global variable x and processes which are not part of P cannot see the information generated by P about x.
- **next** P executes process P in the next time unit (unit-delay)
- **unless** c **next** P executes P iff c cannot be entailed by the constraint store *in the current time unit*
- $!P$ executes P in all time units from the current one on. It can be viewed as $P||$**next** $P||$**next next** $P||...$
- $\star P$ represents unbounded but finite delays, i.e P eventually will be executed. This process can be viewed as $P +$ **next** $P +$ **next next** $P...$**next** ^nP where n is a finite natural number.

Two new operators (ρP and $\star_\rho P$) will be introduced in section 3.1 to express stochastic behavior. They will be illustrated in the model of the lambda switch.

2.1.4 Rules of internal reduction

In this section we show the operational semantics of `ntcc` by giving reduction rules for each process. These rules will help us to understand how `ntcc` processes interact with each other until they reach a quiescent point. Recall that when this state is reached, another time units is created with an empty constraint store and the *residual* process. For a complete description of `ntcc` semantics refer to [6]. Reduction rules are based on *configurations*. A configuration $\langle P, d\rangle$ is composed of a `ntcc` process P and a store d.
For **tell** processes we have:

$$TELL \frac{}{\langle \textbf{tell } c, d\rangle \rightarrow \langle \textbf{skip}, d \wedge c\rangle}$$

where **skip** is the empty process. This reaction says that a **tell** process adds information (a constraint) to the constraint store d.

In **when** c **do** P processes the rule is as follows:

$$SUM \frac{}{\langle \sum_{i\in I} \textbf{when } c_i \textbf{ do } P_i, d\rangle \rightarrow \langle P_j, d \wedge c\rangle} if \quad d \models c_j, \quad j \in I$$

It means that a particular process P_j is non-deterministically chosen for execution among all those whose guard (c_i) can be entailed from the current store d.
For parallel composition we have:

$$PAR \frac{\langle P, c\rangle \rightarrow \langle P', d\rangle}{\langle P||Q, c\rangle \rightarrow \langle P'||Q, d\rangle}$$

It says that if P evolves to P', then the same transition can occur if we execute P in parallel with some process Q. Parallel composition is commutative.
For **unless** c **next** P processes:

$$UNLESS \frac{}{\langle \textbf{unless } c \textbf{ next } P, d\rangle \rightarrow \langle \textbf{skip}, d\rangle} if \quad d \models c$$

The rule says that nothing is done when c is entailed by the store.
Finally, the rule for star processes is:

$$STAR \frac{}{\langle \star P, d\rangle \rightarrow \langle \textbf{next } ^nP, d\rangle} if \quad n \geq 0$$

It means that process P will be run in the (undetermined) future.

The above rules define so-called *internal* transitions. In addition to these, `ntcc` defines an *observable* transition which is the one that goes from one time unit to the next. At the end of a time unit the resulting store can be observed by the environment. Then, processes contained in **next** constructs are scheduled for the next time unit. This include those defined by **unless** processes whose guard cannot be entailed from the current store (see [6] for details).

2.2 Linear-temporal Logic in ntcc

ntcc can be used to verify properties over timed systems. It provides for this a linear temporal logic in which temporal properties over infinite sequences of constraints can be stated [6]. The syntax of this logic is as follows:

$$A, B, ... : c \mid A \dot{\Rightarrow} A \mid \dot{\neg} A \mid \dot{\exists}_x A \mid \circ A \mid \Diamond A \mid \Box A$$

c is a constraint. $\dot{\Rightarrow}$, $\dot{\neg}$ and $\dot{\exists}_x$ represent the linear-temporal logic implication, negation and existential quantification, respectively [4]. These symbols should not be confused with their counterpart in the constraint system (i.e $\Rightarrow$, $\neg$ and $\exists$). Symbols $\circ$, $\Box$ and $\Diamond$ denote the temporal operators *next*, *always* and *eventually*.

The interpretation structures of formulae in this logic are infinite sequences of states [4]. In ntcc , states are replaced by constraints. Given the set C of constraints in the constraint system, let $\alpha \in C^\infty$ be an infinite sequence of constraint and $\alpha(i)$ the $i-th$ element of α. We say that $\alpha \in C^\infty$ is a model of (or that ii satisfies) A, notation $\alpha \models A$, if $\langle \alpha\ ,\ 1\rangle \models A$ where:

$$\begin{array}{lll}
\langle \alpha\ ,\ i\rangle \models c & iff & \alpha(i) \models c \\
\langle \alpha\ ,\ i\rangle \models \dot{\neg} A & iff & \langle \alpha\ ,\ i\rangle \not\models\ A \\
\langle \alpha\ ,\ i\rangle \models A_1 \dot{\Rightarrow} A_2 & iff & \langle \alpha\ ,\ i\rangle \models A_1\ implies \langle \alpha\ ,\ i\rangle \models A_2 \\
\langle \alpha\ ,\ i\rangle \models \circ A & iff & \langle \alpha\ ,\ i+1\rangle \models A \\
\langle \alpha\ ,\ i\rangle \models \Box A & iff & \forall_{j\geq i} \langle \alpha\ ,\ j\rangle \models A \\
\langle \alpha\ ,\ i\rangle \models \Diamond A & iff & \exists_{j\geq i}\ s.t. \langle \alpha\ ,\ j\rangle \models A \\
\langle \alpha\ ,\ i\rangle \models \dot{\exists}_x A & iff & there\ is\ an\ x-variant\ \alpha'\ of\ \alpha\ s.t.\ \langle \alpha'\ ,\ i\rangle \models A
\end{array} \quad (1)$$

In the last expression, d and α' are $x-$variants of c and α , respectively, if they are the same except for the information about x.

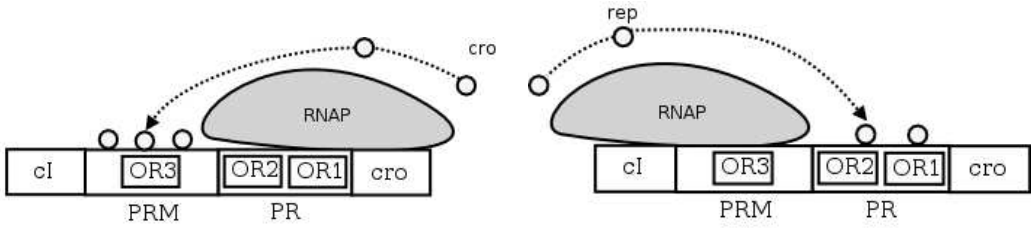
In [6] a proof system is built on the top of this logic. Given a process P and a formula A, a proof of $P \models A$ can be obtained by following a set of inference rules. Nevertheless, we are interested in proving properties with probabilistic statements such as "*The concentration of component c will eventually become 0 with probability* ρ". In section 3.3 we provide an inference system to prove these kind of properties.

2.3 Lambda Switch

In this section we give a brief description of a biological system (called the λ switch) that we model in section 4 using our proposed extension of ntcc . For additional details see [3] or [1].
Bacteriophage λ is a virus that infects the *Escherichia coli* bacterium. As we will see, this biological system exhibits cooperativity relationships and non-deterministic behavior. When the virus injects its genome into the bacteria, there are two states that the bacteria can reach: (1)*lytic growth* in which the virus produces new viruses and (2) *lysogeny* in which the viral genome is passed to new generations in a passive way.
The switching between states is determined by processes in a region of the virus genome called the λ switch (see figure 1). In this switch there are two promoter regions called PRM and PR where production of *rep* and *cro* proteins, respectively, take place. The *lytic growth* state is characterized by a high concentration of *cro* proteins whereas *lysogeny* is characterized by a high concentration of *rep*.

Promoters are overlapped by three regions (binding sites) called $OR1$, $OR2$ and $OR3$. Region $OR1$ exhibits a high affinity for *rep* and a low affinity for *cro*. On the other hand, $OR3$ exhibits a high affinity for *cro* and a low affinity for *rep*.

In *lysogeny*, *rep* proteins usually bind $OR1$ and $OR2$. When $OR1$ is bound by *rep*, $OR2$ affinity for *rep* increases. This is a cooperation relation between bindings at different sites. On the other hand, $OR3$ and the promoter PRM are usually vacant but eventually bound by the polymerase $RNAP$. When this occurs, the transcription of the gene cI starts and new instances of *rep* proteins are produced. When $OR1$ is not vacant, binding of $RNAP$ to PR is inhibited, stopping in this way the production of *cro*. Another cooperation relationship is present in the lambda switch: since PRM is a weak promoter, when $OR2$ is bound by *rep*, *rep* cooperates with $RNAP$ and more frequent transcriptions of gene cI happen. It implies that more *rep*



(a) Lytic growth (b) Lysogeny

Figure 1: Lambda switch states

proteins are produced thus increasing the chance of maintaining the *lysogeny* state.
Lysogeny state is maintained until an environmental signal turns the switch to the *lytic growth*. This process is called *Induction* and it occurs with very low probability. In this state the concentration of *rep* decreases dramatically and so $RNAP$ has the chance to bind to $OR1$, that is now vacant most of the time. PR, a stronger promoter than PRM, starts the transcription of gene *cro* and thus new instances of *cro* are produced. Because $OR3$ has a high affinity for *cro*, it is bound by this protein avoiding $RNAP$ bindings to PRM.
This biological system has been successfully modeled in [3] by using the π calculus [8] with stochastic behavior [7]. However, we believe that notions such as cooperation and the effect of distance (i.e reactions may occur depending on the distance between molecules) can be expressed in a more straightforward manner by using the notion of constraint. We also introduce some additional details that were left aside in the model in [3].

3 Introducing Stochastic Behavior in `ntcc`

In this section we introduce two new operators in `ntcc` for modeling stochastic behavior. We also propose an extension of the `ntcc` linear-temporal logic that can be used to prove properties involving probabilistic statements.

3.1 $\star_\rho P$ Processes

Star processes are used to express eventually in `ntcc` . However, we should be able to differentiate between two processes that eventually occur with different probabilities. For example, the *induction* process in the lambda switch occurs eventually but with a very low probability, while bindings between *rep* and region $OR3$ are quite frequent in *lysogeny*. We propose the constructor $\star_\rho P$ in which P eventually occurs with probability ρ. Operationally:

$$STAR_\rho \frac{}{\langle \star_\rho P, d\rangle \rightarrow \langle \mathbf{next}^n P, d\rangle} if\ n \geq 0 \wedge \Phi(\rho) = 1$$

where $\Phi : \Re \in [0,1] \rightarrow bool$. Φ can be computed by generating pseudo-random numbers following a binomial distribution with probability ρ and 1 as number of events. If $\Phi(\rho) = 1$ we say that the process will be executed, otherwise it will not.

This operator can be expressed as combinations of the original ones:

$$\star_\rho P \stackrel{def}{\equiv} \mathbf{local}\ x\ \mathbf{in}\ \star(\mathbf{tell}\ x = \Phi(\rho) || \mathbf{when}\ x = 1\ \mathbf{do}\ P)$$

by adding the function Φ to the signature of the constraint system.

3.2 ${}_\rho P$

The previous constructor operates over an arbitrary time interval. We are interested in expressing processes that occur in a single time unit but with a probability ρ. Those processes, denoted ${}_\rho P$, can be expresses as follows:

$$ {}_\rho P \equiv \textbf{local } x \textbf{ in tell } (x = \Phi(\rho))||\textbf{when } x = 1 \textbf{ do } P \tag{2}$$

Operationally:

$$RHOP\frac{}{\langle {}_\rho P, d\rangle \rightarrow \langle P, d\rangle} if\ \Phi(\rho) = 1$$

3.3 Stochastic parameters in the linear-temporal logic

To prove properties such as "*The concentration of rep will eventually become 0 with probability* ρ" we change the structure of formulae by adding probabilities, i.e formulae will be tuples $\langle A\ ,\ \rho\rangle$ where A is a formula in the `ntcc` lineal-temporal logic. The new operators are as follows:

$$A', B', ... = \langle c, \rho\rangle \mid A' \stackrel{\circ}{\Rightarrow} A' \mid \stackrel{\circ}{\neg} A' \mid \stackrel{\circ}{\exists}_x A' \mid \circ A',\ \mid \Diamond A' \mid \Diamond A'$$

Probability ρ should not be confused with some notion of *degree of validity* of the formula. This probability refers to the occurrence of events (time). For example, the formula $\langle c < 3, 0.5\rangle$ expresses that with a probability of 0.5 constraint $c < 3$ will be asserted.
The semantics of our new logic is as follows:

$$\begin{array}{lll}
\langle \alpha\ ,\ i\rangle \models \langle c\ ,\ \rho\rangle & iff & \alpha(i) \models \langle c\ ,\ \rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \stackrel{\circ}{\neg}\langle A\ ,\ \rho\rangle & iff & \langle \alpha\ ,\ i\rangle \not\models\ \langle A\ ,\ 1-\rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \langle A_1\ ,\ \rho_1\rangle \stackrel{\circ}{\Rightarrow} \langle A_2\ ,\ \rho_2\rangle & iff & \langle \alpha\ ,\ i\rangle \models \langle A_1\ ,\ \rho_1\rangle\ implies \langle \alpha\ ,\ i\rangle \models \langle A_2\ ,\ \rho_2\rangle \\
\langle \alpha\ ,\ i\rangle \models \circ\langle A\ ,\ \rho\rangle & iff & \langle \alpha\ ,\ i+1\rangle \models \langle A\ ,\ \rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \Box\langle A\ ,\ \rho\rangle & iff & \forall_{j\geq i}\langle \alpha\ ,\ j\rangle \models \langle A\ ,\ \rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \Diamond\langle A\ ,\ \rho\rangle & iff & \exists_{j\geq i}\ s.t. \langle \alpha\ ,\ j\rangle \models \langle A\ ,\ \rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \stackrel{\circ}{\exists}_x\langle A\ ,\ \rho\rangle & iff & there\ is\ an\ x-variant\ \alpha'\ of\ \alpha\ s.t.\ \langle \alpha'\ ,\ i\rangle \models \langle A\ ,\ \rho\rangle
\end{array} \tag{3}$$

where $\alpha(i) \models \langle c\ ,\ \rho\rangle$ is defined as: $\langle c_1\ ,\ \rho_1\rangle \models \langle c_2\ ,\ \rho_2\rangle$ iff c_1 entails c_2 in the constraint system and $\rho_2 \leq \rho_1$.

3.3.1 Inference System

We extend the inference system proposed in [6] with inferences rules taking into account the new form of formulae and the probabilistic operators ${}_\rho P$ and $\star_\rho P$. The rules are as follows:

$$LTELL : \textbf{tell } c \vdash \langle c\ ,\ 1.0\rangle \tag{4}$$

$$LPAR : \frac{P \vdash \langle A\ ,\ \rho\rangle \quad Q \vdash \langle B\ ,\ \rho_2\rangle}{P||Q \vdash \langle A\ ,\ \rho\rangle \stackrel{\circ}{\wedge} \langle B\ ,\ \rho_2\rangle} \tag{5}$$

i.e. the parallel execution of two process satisfies the conjunction of the formulae of each process.

$$LLOC : \frac{P \vdash \langle A\ ,\ \rho\rangle}{\textbf{local } x \textbf{ in } P \vdash \stackrel{\circ}{\exists}_x\langle A\ ,\ \rho\rangle} \tag{6}$$

$$LNEXT : \frac{P \vdash \langle A\ ,\ \rho\rangle}{\textbf{next } P \vdash \circ\langle A\ ,\ \rho\rangle} \tag{7}$$

$$LUNL : \frac{P \vdash \langle A\ ,\ \rho\rangle}{\textbf{unless } c \textbf{ next } P \vdash \langle c\ ,\ 1.0\rangle \stackrel{\circ}{\vee} \circ\langle A\ ,\ \rho\rangle} \tag{8}$$

$$LCONS : \frac{P \vdash \langle A\ ,\ \rho\rangle}{P \vdash \langle B\ ,\ \rho_2\rangle}\ if\ \langle A\ ,\ \rho\rangle \stackrel{\circ}{\Rightarrow} \langle B\ ,\ \rho_2\rangle \tag{9}$$

$$LSTAR : \frac{P \vdash \langle A\ ,\ \rho\rangle}{\star P \vdash \Diamond\langle A\ ,\ \rho\rangle} \tag{10}$$

$$LSUM : \frac{\forall i \in I \quad P_i \vdash \langle A_i\ ,\ \rho_i\rangle}{\sum_{i\in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i \vdash \overset{\circ}{\bigvee}_{i\in I}(\langle c_i\ ,\ 1\rangle \overset{\circ}{\wedge} \langle A_i\ ,\ \rho_i\rangle) \overset{\circ}{\vee} \overset{\circ}{\bigwedge}_{i\in I} \overset{\circ}{\neg}\langle c_1\ ,\ 1.0\rangle} \tag{11}$$

$$PPRO : \frac{P \vdash \langle A\ ,\ \rho\rangle}{{}_{\rho_2}P \vdash \langle A\ ,\ \rho \times \rho_2\rangle} \tag{12}$$

The previous equation represents the inference rule for ρP processes. Since the probability ρ and the probability ρ_2 are independent, the probability of both occurrences is $\rho \times \rho_2$.
Finally, the rule for the $\star_\rho P$ process is:

$$PSTAR : \frac{P \vdash \langle A\ ,\ \rho\rangle}{\star_{\rho_2} P \vdash \Diamond\langle A\ ,\ \rho \times \rho_2\rangle} \tag{13}$$

4 Modeling the λ-Switch with the stochastic ntcc extension

Recall that our objective is to use concurrent calculi to faithfully model biological systems. We test the appropriateness for this task by constructing a (somewhat) detailed model of the lambda-switch using the extended calculus defined above. We use *process definition* constructs of the form $\mathbf{PROCESS(x)} \overset{def}{\equiv} P$ that do not formally belong to the calculus. These, however, can be easily defined in terms of the standard calculus constructs (see [6]).

4.1 REP and CRO Protein Control

In our model production of *rep* proteins is controlled by 3 extended calculus processes: (1) Induction, that reduces the concentration of *rep* thus switching the system to a *lytic growth* state, (2) a process supervising that the concentration of *rep* does not exceed a given threshold and (3) a process that increases the concentration of *rep* as a result of the expression of gene cI.

The induction process (see section 2.3), activated by environment signals that are at present little understood, causes a strong reduction of **rep** proteins and therefore switching to a *lytic growth* state. The occurrence probability for this process is very low. By using $\star_\rho P$ processes we can model this fact as:

$$\mathbf{INDUCTION}(\rho_{\mathbf{ind}}) \overset{def}{\equiv} \star_{\rho_{ind}}(\mathbf{tell}\ reset\_rep_c) \tag{14}$$

Eventually, under the probability ρ_{ind}, this process reduces the concentration of *rep* to 0 (see $reset\_rep_c$ in equation 21).
The following process avoids a negative feedback by controlling the concentration of *rep* proteins. This represents the situation in which there are so many *rep* proteins floating around that even $OR3$ will get bound to *rep* thus inhibiting gene cI expression. The process inhibits the production of *rep* when concentration reaches $rep_{threshold}$.

$$\begin{array}{l}\mathbf{REPTHOLD}(\mathbf{rep_{threshold}}) \overset{def}{\equiv} \\ \mathbf{when}\ C_{rep} > rep_{threshold}\ \mathbf{do\ next\ tell}\ inhibit_{rep}\end{array} \tag{15}$$

4.2 Gene transcription

As mentioned before, PRM is a weaker promoter than PR . This means that production of *rep* takes more time than production of *cro* once $RNAP$ binds to PRM (resp. PR). Sometimes no *rep* protein is produced at all when the binding occurs, i.e $RNAP$ falls off without transcribing gene cI.

We model the gene transcription at the PR promoter as follows:

$$\mathbf{CROtrans} \overset{def}{\equiv} \mathbf{when}\ pr_{bound}\ \mathbf{do}\ {}_{\rho_{crotrans}}(\mathbf{tell}\ inc\_cro_c) \tag{16}$$

where predicate pr_{bound} is true iff $RNAP$ is binding PR . Constraint inc_cro_c causes a new (higher) value for concentration of *cro* proteins to be asserted in the next time unit (see equation 21).
Modeling gene transcription at PRM is more difficult because the probability of transcription may vary according to the presence or absence of *rep* at $OR2$. When $OR2$ is bound by *rep* , the probability of gene *cI* transcription at PRM is higher and in consequence higher is also the probability of producing more *reps*. Equation 17 models gene transcription at PRM :

$$\begin{array}{c}\mathbf{CItrans}(\rho_{high},\rho_{low}) \stackrel{def}{\equiv} \mathbf{when}\ \neg inhibit_{rep} \wedge prm_{bound}\ \mathbf{do\ local}\ p\ \mathbf{in}\\ (\mathbf{when}\ or2\_bound\_rep\ \mathbf{do\ tell}\ p=\rho_{high}\\ +\ \mathbf{when}\ or2\_bound\_cro \vee or2\_vacant\ \mathbf{do}\ p=\rho_{low})\\ ||\quad {}_p(\mathbf{tell}\ inc\_rep_c)\end{array} \tag{17}$$

where cooperativity is modelled by means a **when** c **do** P process that chooses between ρ_{high} and ρ_{low} in order to execute **tell** $inc(rep_c)$ with the right probability. Notice that synchronization is guaranteed by the use of constraints: process ${}_p(\mathbf{tell}\ inc\_rep_c)$ blocks until the value of p is known.

4.3 Operator Regions

In what follows we model the behavior of the $OR1, OR2$ y $OR3$ binding sites. Since each operator has different affinities for *rep* and *cro* , ${}_\rho P$ type processes are needed. $OR2$ and $OR1$ exhibit another cooperativity relationship: $OR2$ increases its affinity for *rep* when $OR1$ is bound by *rep* . The following equations model the operator regions in the lambda switch:

$$\begin{array}{l}\mathbf{OR1} \stackrel{def}{\equiv}\\ \quad \mathbf{when}\ or1\_unbound\ \mathbf{do}\\ \qquad\qquad \mathbf{when}\ rep_c>0\ \mathbf{do}\ {}_{\rho_{or1rep}}(\mathbf{next\ tell}\ (or1\_rep\_bound \wedge dec\_rep_c))\\ \qquad\qquad +\ \mathbf{when}\ cro_c>0\ \mathbf{do}\ {}_{\rho_{or1cro}}(\mathbf{next\ tell}\ (or1\_cro\_bound \wedge dec\_cro_c))\\ \quad \mathbf{when}\ or1\_rep\_bound\ \mathbf{do}\ {}_{1.0-\rho_{or1rep}}(\mathbf{next\ tell}\ (or1\_unbound \wedge inc\_rep_c))\\ \quad +\ \mathbf{when}\ or1\_cro\_bound\ \mathbf{do}\ {}_{1.0-\rho_{or1cro}}(\mathbf{next\ tell}\ (or1\_unbound \wedge inc\_cro_c))\end{array} \tag{18}$$

$$\begin{array}{l}\mathbf{OR2} \stackrel{def}{\equiv} \mathbf{local}\ or2rep\ \mathbf{in}\\ \quad \mathbf{when}\ or2\_unbound\ \mathbf{do}\\ \qquad\qquad \mathbf{when}\ rep_c>0\ \mathbf{do}\ {}_{\rho_{or2rep}}(\mathbf{next\ tell}\ (or2\_rep\_bound \wedge dec\_rep_c))\\ \qquad\qquad +\ \mathbf{when}\ cro_c>0\ \mathbf{do}\ {}_{\rho_{or2cro}}(\mathbf{next\ tell}\ (or2\_cro\_bound \wedge dec\_cro_c))\\ \qquad\qquad +\ \mathbf{when}\ \neg or1\_bound\_rep\ \mathbf{do\ tell}\ (\rho or2rep=\rho or2rep\_low)\\ \quad\ \mathbf{when}\ or2\_rep\_bound\ \mathbf{do}\ {}_{1.0-\rho_{or2rep}}(\mathbf{next\ tell}\ (or2\_unbound \wedge inc\_rep_c))\\ \quad\ \mathbf{when}\ or2\_cro\_bound\ \mathbf{do}\ {}_{1.0-\rho_{or2cro}}(\mathbf{next\ tell}\ (or2\_unbound \wedge inc\_cro_c))\\ \|(\mathbf{when}\ or1\_bound\_rep\ \mathbf{do\ tell}\ (\rho or2rep=\rho or2rep\_high)\end{array} \tag{19}$$

$$\begin{array}{l}\mathbf{OR3} \stackrel{def}{\equiv}\\ \mathbf{when}\ or3\_unbound\ \mathbf{do}\\ \qquad\qquad \mathbf{when}\ rep_c>0\ \mathbf{do}\ {}_{\rho_{or3rep}}(\mathbf{next\ tell}\ (or3\_rep\_bound \wedge dec(rep_c))\\ \qquad\qquad +\ \mathbf{when}\ cro_c>0\ \mathbf{do}\ {}_{\rho_{or3cro}}(\mathbf{next\ tell}\ (or3\_cro\_bound \wedge dec(cro_c))\\ \quad \mathbf{when}\ or1\_rep\_bound\ \mathbf{do}\ {}_{1.0-\rho_{or3rep}}(\mathbf{next\ tell}\ (or3\_unbound \wedge inc(rep_c))\\ \quad +\ \mathbf{when}\ or1\_cro\_bound\ \mathbf{do}\ {}_{1.0-\rho_{or3cro}}(\mathbf{next\ tell}\ (or3\_unbound \wedge inc(cro_c))\end{array} \tag{20}$$

where $\rho or1rep$,$\rho or1cro$,$\rho or2rep\_high$,$\rho or2rep\_low$,$\rho or3rep$ and $\rho or3cro$ are the probabilities (affinities) of each binding site w.r.t *rep* and *cro*. Constraints dec_rep_c and dec_cro_c will cause decrementing of the concentration of *rep* (resp. *cro*) in the environment (see equation 21). In the above processes, when the operator is unbound (i.e vacant) and *reps* or *cros* are available in the environment, eventually the protein (i.e *rep* or *cro*) binds the operator and consequently also decreases the protein concentration in the environment. Finally, equation 21 controls the concentration of *rep* and *cro* for the next time unit:

$$\begin{array}{l}\mathbf{CONCCTR} \stackrel{def}{\equiv}\\ (\mathbf{when}\ reset\_rep_c\ \mathbf{do\ next\ tell}\ rep_c=0 + \mathbf{when}\ inc\_rep_c \wedge \neg reset\_rep_c\ \mathbf{do\ next\ tell}\ (rep_c=rep_c+1)+\\ \mathbf{when}\ dec\_rep_c \wedge \neg reset\_rep_c\ \mathbf{do\ next\ tell}\ (rep_c=rep_c-1))||\\ (\mathbf{when}\ inc\_cro_c\ \mathbf{do\ next\ tell}\ (cro_c=cro_c+1) + \mathbf{when}\ dec\_cro_c\ \mathbf{do\ next\ tell}\ (cro_c=cro_c-1))\end{array} \tag{21}$$

4.4 RNAP binding

Equations 16 and 17 depend on bindings between $RNAP$ and promoters PR and PRM , respectively. The behavior of $RNAP$ has two components: (1) when $RNAP$ is binding PR (resp PRM), there is a probability of $1.0-\rho rnap\_pr$ (resp. $1.0-\rho rnap\_prm$) of falling off thus interrupting gene transcription. And (2) when promoters are unbound, $RNAP$ may bind to PR (resp PRM) if $OR1$ (resp. $OR3$) is vacant, with probability $rnap\_pr$ (resp. $rnap\_prm$). Equation 22 models this fact:

$$\begin{array}{l}\textbf{RNAPCTR} \stackrel{def}{\equiv} \\ \textbf{when } rnap\_unbound \textbf{ do } (\\ \textbf{when } or1\_unbound \textbf{ do } {}_{rnap\_pr}(\textbf{next tell } (or1\_rnap\_bound \wedge pr\_bound)) \\ \quad + \textbf{when } or3\_unbound \textbf{ do } {}_{rnap\_prm}(\textbf{next tell } (or3\_rnap\_bound \wedge prm\_bound)) \\ +\textbf{when } rnap\_bound \textbf{ do } (\\ \textbf{when } or1\_rnap\_bound \textbf{ do } {}_{1.0-rnap\_pr}(\textbf{next tell } or1\_unbound) \\ +\textbf{when } or3\_rnap\_bound \textbf{ do } {}_{1.0-rnap\_prm}(\textbf{next tell } or3\_unbound))\end{array} \tag{22}$$

With equations 14 to 22 we can describe the overall lambda-switch system as follows:

$$\begin{array}{l}\lambda - \textbf{PROC}(\rho\_ind, rep\_thr, \rho or1rep, \rho or2rep\_high, \rho or2rep\_low, \rho or3rep, \\ \qquad \rho or1cro, \rho or2cro, \rho or3cro, \rho crotrans, \rho citrans\_high, \rho citrans\_low) \stackrel{def}{\equiv} \\ \textbf{local } rep_c(0), cro_c(0), pr_{bound}, prm_{bound} inhibit_{rep} \\ \quad or1\_bound\_rep, or1\_bound\_cro, or2\_bound\_rep, or2\_bound\_cro, \\ \quad or2\_bound\_rep, or2\_bound\_cro, or1\_unbound, or2\_unbound, or3\_unbound, \\ \quad rnap\_bound, rnap\_unbound, or1\_bound\_rnap, or2\_bound\_rnap, or3\_bound\_rnap \\ \quad \rho_{or1rep}, \rho_{or2rep}, \rho_{or3rep}, \rho_{or1cro}, \rho_{or2cro}, \rho_{or3cro} \\ \textbf{in} \\ \quad INDUCTION(\rho\_ind)||!REPTHOLD(rep\_thr)||!CROtrans||!CItrans|| \\ \quad !OR1||!OR2||!OR3||!RNAPCTR||!CONCCTR\end{array} \tag{23}$$

This equation defines the stochastic parameters of the model and then execute concurrently the processes needed to control the behavior of the lambda switch.

4.5 Proving temporal properties of the λ switch system

In this section we show proofs for two properties the lambda switch system must satisfy:

- *Eventually, with probability ρ (a very low probability in this case) the concentration of rep proteins drops to zero .*
- *If $OR1$ and $OR2$ are bound by reps and $OR3$ is bound by $RNAP$, a new instance of rep will eventually be produced (i.e. the protein concentration will be incremented) with probability ρ_{ci_high}. Recall that when $OR2$ is bound by rep, the rate of transcription of gene cI is incremented because of the cooperativity relationship between $OR2$ and PRM .*

In order to proof the first property, we star with the definition of the overall system:

$$LPAR : \frac{INDUCTION \vdash \langle A\ ,\ \rho\rangle \quad OR1 \vdash \langle A_2\ ,\ \rho_2\rangle\ ...\ RNAPCTR \vdash \langle A_n\ ,\ \rho_n\rangle}{\lambda - PROC \vdash \langle A\ ,\ \rho\rangle \overset{\circ}{\wedge} \Box\langle A_2\ ,\ \rho_2\rangle ... \Box \overset{\circ}{\wedge} \langle A_n\ ,\ \rho_n\rangle} \tag{24}$$

Notice that we use the temporal *always* operator because processes are replicated (i.e. use the "!" prefix) in the definition of $\lambda - PROC$.

By using $LCONS$ in equation 24 we get:

$$LPAR : \overline{\lambda - PROC \vdash \langle A\ ,\ \rho\rangle} \tag{25}$$

The $INDUCTION$ process satisfies the formula $\langle A\ ,\ \rho\rangle$. Now, we are going to find out the structure of A:

$$PSTAR : \frac{\textbf{tell}\ reset(rep_c) \vdash \langle reset(rep_c)\ ,\ 1.0\rangle}{\star_{\rho\_ind}\textbf{tell}\ reset(rep_c)||INDUCTION \vdash \Diamond\langle reset(rep_c)\ ,\ \rho\_ind\rangle} \tag{26}$$

As $INDUCTION \stackrel{def}{\equiv} \star_{\rho\_ind}\textbf{tell}\ reset(rep_c)||INDUCTION$ (omitting the local hiding), we can affirm that $\lambda - PROC$ satisfies the property "*eventually under a probability* ρ_ind the concentration of *rep* will be zero" , i.e, $\Diamond\langle reset(rep_c)\ ,\ \rho\_ind\rangle$.

To prove the second property we star from the definition of $\lambda - PROC$ and then use the definitions of $CItrans$ and $RNAPCTR$:

$$LPAR : \frac{RNAPCTR \vdash \langle A\ ,\ \rho\rangle \quad CITRANS \vdash \langle A_2\ ,\ \rho_2\rangle\ ...}{\lambda - PROC \vdash \Box\langle A\ ,\ \rho\rangle \stackrel{\circ}{\wedge} \Box\langle A_2\ ,\ \rho_2\rangle ...} \tag{27}$$

Using the definition of $CITRANS$ we get:

$$LSUM : \frac{\textbf{tell}\ p = \rho\_high \vdash \langle p = \rho\_high\ ,\ 1.0\rangle \quad \textbf{tell}\ p = \rho\_low \vdash \langle p = \rho\_low\ ,\ 1.0\rangle}{\begin{array}{c}\textbf{when}\ or2\_bound\_rep\ \textbf{do tell}\ p = \rho\_high + \textbf{when}\ or2\_vacant\ \textbf{do tell}\ p = \rho\_low \vdash \\ \langle or2\_bound\_rep \dot{\wedge} p = \rho\_high\ ,\ 1.0\rangle \stackrel{\circ}{\vee} \langle or2\_vacant \dot{\wedge} p = \rho\_low\ ,\ 1.0\rangle \stackrel{\circ}{\vee} \\ \langle \dot{\neg} or2\_bound\_rep \dot{\wedge} \dot{\neg} or2\_vacant\ ,\ 1.0\rangle\end{array}} \tag{28}$$

Since the premise is the formula $\langle or1\_bound\_rep \dot{\wedge} or2\_bound\_rep \dot{\wedge} or3\_bound\_rnap\ ,\ 1.0\rangle$ we can use $LCONS$ as follows:

$$LCONS : \frac{\langle or1\_bound\_rep \dot{\wedge} or2\_bound\_rep \dot{\wedge} or3\_bound\_rnap\ ,\ 1.0\rangle}{\begin{array}{c}\textbf{when}\ or2\_bound\_rep\ \textbf{do tell}\ p = \rho\_high + \textbf{when}\ or2\_vacant\ \textbf{do tell}\ p = \rho\_low \vdash \\ \langle p = \rho\_high\ ,\ 1.0\rangle\end{array}} \tag{29}$$

Finally, by using $PPRO$ we can verify that $CITRANS$ satisfies the following formula:

$$\frac{\textbf{when}\ or2\_bound\_rep\ \textbf{do tell}\ p = \rho\_high + \textbf{when}\ or2\_vacant\ \textbf{do tell}\ p = \rho\_low \vdash \langle p = \rho\_high\ ,\ 1.0\rangle}{\textbf{when}\ ...\ \textbf{do}\ ... + \textbf{when}\ ...\ \textbf{do}\ ...||\ {}_p\_high(\textbf{tell}\ inc(rep_c)) \vdash \langle inc(rep_c)\ ,\ \rho\_high\rangle} \tag{30}$$

Equation 30 says that new *rep*s will appear with a probability ρ_high verifying the cooperative behavior between $OR2$ and PR .

5 Concluding remarks and Future Work

We have given orthogonal extensions of `ntcc` for modeling stochastic behavior of processes. In particular, we proposed two new operators: $\star_\rho P$ and ${}_\rho P$. The first one expresses that P is eventually executed with probability ρ and the second one that P is executed in the current time unit with probability ρ. We showed that both can be expressed in terms of existing `ntcc` constructs by including in the signature of the underlying constraint system of `ntcc` a probabilistic function $\Phi(\rho) : [0, 1] \rightarrow bool$ following a binomial distribution. The syntax and semantics associated to the temporal logic and proof system of `ntcc` were modified accordingly to account for the new stochastic constructs. We have shown that the proposed extension provides enough expressiveness to faithfully model a non trivial stochastic system. Using this new stochastic non-deterministic calculus we were able to provide a model of a biological system called the lambda switch that is both simpler and more complete than models previously proposed based on the stochastic π-calculus.

Finally, an inference system to prove probabilistic properties in the calculus was provided. This inference system is built on an `ntcc` linear-temporal logic extension by adding probabilities to each formula. For each process construction including our new processes $\star_\rho P$ and ${}_\rho P$, we defined an inference rule to proof if a process P satisfies an specification (i.e a formula in the logic). We showed the use of this system by proving two properties in our lambda switch model.

In the short term we plan to implement a `ntcc` stochastic processes simulator to better visualize process behavior. We will build it up from a `ntcc` framework previously developed by our group ([5]) . This will allow us to trace the evolution of protein concentrations so as to assess our model w.r.t. existing biological data for the lambda switch. Additionally, we expect to use our proof system to verify properties of relevance to biologists. Adapting or implementing a new automatic theorem-prover will help us in this task.
We also plan to construct reasonably complete models of some other biological systems such as the N_A *pump* [2].

References

[1] Adam Arkin, John Rossb, and Harley H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. In *Genetics*, 1998.

[2] D. Besozzi and G. Ciobanu. A p system description of the sodium-potassium pump. In *Workshop on Membrane Computing*, 2004.

[3] Celine Kuttler, Joachim Niehren, and Ralf Blossey. Gene regulation in the pi calculus: Simulating cooperativity at the lambda switch. In *Bio-CONCUR 2004*, 2004.

[4] Z. Manna and A. Penueli. *The Temporal Logic of Reactive and Concurrent Systems, Sepecification.* Springer, 1991.

[5] Pilar Munoz and Andres Hurtado. Programming robot devices with a timed concurrent constraint programming. In *Principles and Practice of Constraint Programming - CP2004. LNCS 3258*, page 803. Springer, 2004.

[6] Mogens Nielsen, Catuscia Palamidessi, and Frank D. Valencia. Temporal concurrent constraint programming: Denotation, logic and applications. In *Special Issue of Selected Papers from EXPRESS'01, Nordic Journal of Computing*, 2001.

[7] C. Priami. Stochastic pi-calculus. In *Computer Journal*, 2004.

[8] J. Parrow R. Milner and D. Walker. A calculus of mobile processes, Parts I and II. *Journal of Information and Computation*, 100:1–77, September 1992.

[9] Aviv Regev, William Silverman, and Ehud Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Pacific Symposium on Biocomputing*, pages 459–470, 2001.

[10] V. Saraswat, R. Jagadeesan, and V. Gupta. Fundation of timed concurrent constraint programming. In *IEEE Symposium on Logic in Computer Science*. IEEE press, 1994.

[11] V. A. Saraswat. *Concurrent Constraint Programming.* The MIT Press, Cambridge, MA, 1993.

Verification of Rewrite Based Specifications using Proof Assistants

Thomas Mailleux Sant'Ana and Mauricio Ayala-Rincón*
Mestrado em Informática and Departamento de Matemática, Universidade de Brasília,
Brasília – D. F., 70900-010, Brasil
mailleux@gmx.net, ayala@mat.unb.br

Abstract

Recent works demonstrate the applicability of rewriting-logic environments for the specification of hardware. When these specification are proved to be correct one can additionally apply them for the simulation, testing and even analysis of the conceived specified hardware. But theorem proving mechanisms are not included as basic/natural components of rewriting-logic environments (such as ELAN, CafeObj and Maude). Even worst, they are not able to handle proofs guided by basic methods of rewriting theory. Consequently, the correctness of these specifications have been done manually. In this work we present a new practical methodology, which is based on a semantically intelligent translation of rewriting-logic specifications in ELAN to theories in the specification language PVS(a well-known proof assistant). This translation includes the generation of lemmas to be checked in order to prove confluence of the original specification, and the possibility of adding lemmas for operational correction and integrity.

Keywords: Rewriting, Rewriting-Logic, Critical Pairs, Proof Assistants, Hardware Verification

1 INTRODUCTION

In [3] the correctness proof of the specification of a speculative processor, presented as a term rewriting system (TRS), was obtained showing that this processor imitates a basic RISC processor and vice-versa. The ELAN specification of the processor was given by a term rewriting system which consists of four sets of specialized rewriting rules: issuing instructions to the buffer, load/store in memory instructions, branch prediction rules and arithmetic and propagation rules. These subset of rules coincide with the ones in [1] except that in [3] rewriting-logical using logical strategies were applied to get a clean discrimination between parts of the processor as well as for implementing different speculative strategies. To prove that each processor imitates the other, it was necessary to verify the convergence of all of the speculative processor's rules, with the exception of the instruction issuing rules. This was done by proving Noetherianity of the rule subset and then applying the Knuth-Bendix-Huet Critical Pair Lemma by hand. This in turn required the exhaustive computation of critical pairs between rewriting rules and subsequent verification of their joinability.

The proposed methodology allows us to mechanically complete these tasks by translating the TRS to proof assistant PVS and using it to assist in the verification process. It is important to stress that the intelligence involved in this translation is not redundant, because no strategies for handling proof based in rewriting basic theory are available in PVS (as well as in most of the known practical proof assistants). Further proofs of integrity constraints of the target specifications are then done in PVS and in the case these proofs are not possible a dynamically reformulation of the rewriting based specification and continuous translation to the language of PVS is possible. In this way verification of complex specifications can be done quickly and safely.

The applicability of the proposed methodology is not restricted to hardware specification, it will work with any rewrite based specification. This is important since specification via rewriting-logic has been shown of practical interest for the modeling and simulation of non standard hardware technologies (for which no commercial tool of correct synthesis is available) such as the innovative reconfigurable systems used in compact and portable systems [4, 2]. Because of this we believe the proposed methodology will be of practical interest in the correct development of reconfigurable technologies.

*Partially supported by the Brazilian research council CNPq.

The remainder of this section briefly discusses the required theoretical basis. Section 2 discusses the goals and outline of the methodology being proposed. Section 3 discusses the translation techniques and choices, it describes why certain routes where used. Section 4 presents one example of application for a large ELAN specification of a basic AX RISC processor [3] and an example of proving the general correctness of a TRS. The last section concludes and presents future work.

1.1 Theoretical Basis

We include the minimal needed notions on rewriting. For detailed presentations see [5, 19].

Rewriting theory has been successfully applied in different areas of computer science as an abstract formalism for assisting the simulation, verification and deduction of complex computational objects and processes. In the context of computer architectures, rewriting theory has been applied as a reasoning tool for hardware design. To review only a reduced set of different approaches in this direction, we mention the work of Kapur who has used his well-known Rewriting Rule Laboratory (RRL) for verifying arithmetic circuits [9, 10] as well as Arvind's group at MIT that dealt with the specification of processors over simple architectures, the rewrite-based description and synthesis of simple logical digital circuits and the description of cache protocols over memory systems [1, 8].

A Term Rewriting System (TRS) is defined as a triplet $\langle R, S, S_0 \rangle$, where S and R are respectively sets of terms and of rewrite rules of the form $l \rightarrow r$ if p being l and r terms and p a predicate, and where S_0 is the subset of allowed initial terms of S. l and r are called the left-hand and right-hand sides of the rule and p it's condition. In the architectural context, terms and rules represent states and state transitions, respectively. A term s can be rewritten or reduced to the term t, denoted by $s \rightarrow t$, whenever there exists a sub-term s' of s that can be transformed according to some rewrite rule into the term s'' such that replacing the occurrence of s' in s with s'' gives t. A term that cannot be rewritten is said to be in normal form. The relation over S given by the previous rewrite mechanism is called the rewrite relation of R and is denoted by $\rightarrow_R$. Its inverse is denoted by ${}_R\leftarrow$ and its reflexive-transitive closure by $\rightarrow_R^*$ and its equivalence closure by $\leftrightarrow_R^*$. The important notions of terminating and confluence properties are defined as usual. These notions correspond to the practical computational aspects as the determinism of processes and their finiteness. A TRS is said to be terminating if there are no infinite sequences of the form $s_0 \rightarrow s_1 \rightarrow \cdots$. A TRS is said to be confluent if for all divergence of the form $s \rightarrow t_1$, $s \rightarrow t_2$, there exists a term u such that $t_1 \rightarrow^* u^* \leftarrow t2$. This ensures that the TRS will produce the same results regardless to the order in which rules are applied to a term. The use of the subset of initial terms S_0, representing possible initial states in the architectural context (which is not standard in rewriting theory), is simply to define what is a "legal" state according to the set of rewrite rules R; i.e., t is a legal term (or state) whenever there exists an initial state $s \in S_0$ such that $s \rightarrow^* t$. Using these notions one can model the operational semantics of algebraic operators and functions.

Although in the pure rewriting context rules are applied in a truly non deterministic manner, for practical use it is necessary to have the control of the ordering in which rules are applied. Thus one needs rewriting jointly with logic, which is known as rewriting-logic [11]. Rewriting-logic extends the pure rewriting paradigm allowing for logical control of the application of the rules by logic strategies. This approach is particularly useful in this context of the specification of processors since they may be adapted for discriminatingly representing in the necessary detail many hardware elements involved in processors [3, 4, 2]. It has also shown use in discriminating between fixed and reconfigurable elements of the innovative reconfigurable architectures used in modern, portable and compact technologies. The use of rewriting-logic allows for a natural and quick conception and simulation of emerging computing paradigms such as *configware* and *morphware* [6] over the conceived reconfigurable architectures which include the sophisticated dynamic reconfiguration mechanisms.

Efforts in the implementation of computational environments and programming languages based on rewriting (matching and substitution) and logic include well-known tools such as ELAN, CafeObj and Maude (see the collection of papers in [11] for recent descriptions of these systems and for a roadmap on rewriting-logic). These systems are useful for implementing and running specifications, but except for the treatment of types they do not include elaborated and natural tools for proving correctness of these specifications, even that based on rewriting basic theory. Maude is an exception to this, having been extended into what is called Full-Maude. This version allowed the creation of model checker, theorem provers, a Knuth-Bendix completion engine, among other applications [11].

When specifying hardware systems, one needs to prove their confluence. This can be achieved by applying the highly used criteria of the Knuth-Bendix-Huet Critical Pair Lemma. Critical pairs are the divergences

formed by the left-side terms of the rules. The lemma states that, for a terminating TRS, one can check confluence simply by showing that all critical pair are joinable, that is, both elements of the pair can be reduced by the rules to the same term. It is a local and effective criterion for verifying convergence of TRS and is the basis of the well-known Knuth-Bendix completion procedure [5, 19].

The proposed methodology translates rewriting based specifications (in ELAN) to the language of the proof assistant PVS, which is the proof assistant used in our experiments. Advantages of PVS include the power of dependent types and higher order logic strategies [15, 18] as well as a lot of accumulated experience in its application in hardware verification [7, 12, 16, 17]

One important aspect of PVS is that type-checking is a integral part of the theorem proving framework, which makes it possible to prove type conditions that are central for correctness proofs of the specifications. *Predicate subtypes* express mathematical ideas such as: naturals being contained in integers, constraints which determine whether a number is even, odd, prime, etc. Predicate subtypes generate a series of *Type Correctness Conditions*, called TCC in PVS, whose demonstrations depend on the extra logic embedded in these predicates instead on conventional type-checking. PVS tries to prove all TCCs automatically and whenever it cannot do so, it generates a proof *obligation* for the user. Any lemma or theorem that depends on that obligation will be considered incomplete until that TCC is proved. This can assist users in detecting errors and problems. For example, consider the lemma shown in equation (1) below. It is not valid because of a possible division by zero. PVS checks all condition required for the lemma and proposes the TCC shown in (2) below. This TCC cannot be proved, so any proof that uses the *div_cancel* lemma will be incomplete. The alternative definition, shown in equation (3) below, does not produce such a TCC, solving the problem. PVS includes a large collection of conditions which are checked against the user supplied specification. And many of these will appear when the specification has flaws.

$$div\_cancel : \textbf{LEMMA}\ \forall x, y \in \mathbb{Z} : \frac{x \times y}{x} = y \tag{1}$$

$$div\_cancel\_TCC1 : \textbf{OBLIGATION}\ \forall (x) : x \neq 0 \tag{2}$$

$$div\_cancel2 : \textbf{LEMMA}\ \forall x \in \mathbb{Z} - \{0\}, y \in \mathbb{Z} : \frac{x \times y}{x} = y \tag{3}$$

As a means to verify the ELAN specification, PVS is used to verify the critical pair joinability. Since PVS does not include strategies for rewriting theory, critical pairs are generated during the translation and the corresponding (joinability) lemmas written in the language of PVS for further verification. In conditional rewriting systems, critical pairs whose conditions are unsatisfiable are considered trivially joinable or simply omitted [14].

A naive objection to the usefulness of the proposed methodology, in the architectural context, is that critical pairs do not arise in practice since in hardware descriptions there is only one path to follow. But important questions when synthesizing hardware are: whether the conceived systems are or not logically correct, as well as whether the expected power consumption and area are efficient. In rewriting-logic, by checking confluence (when it is possible) of a specification, one verifies that all possible computations give the same answer and by logical strategies one can decide (and compare) the best paths to be followed to process a task. The proposed methodology can aid in the conception of efficient dedicated architectures and adequate reconfigurations over reconfigurable architectures. Something of great interest when we consider experiences on using rewriting-logic to model, simulate and even analyze specification such as: speculative processors, efficient dynamically reconfigurable architectures to compute the Fast Fourier Transform [4] and reconfigurable systolic array for general dynamic programming based solutions of relevant problems over molecular sequences such as local and global sequence alignment, approximate pattern matching and longest common subsequence [2].

2 METHODOLOGY DESCRIPTION AND GOALS

The main goal is to provide a methodology to transform term rewrite specifications in ELAN to the language of the proof assistant PVS where one can prove correctness and integrity properties. The transferring and work with PVS allows formal verification of the ELAN specification, including checking types, generating and checking the joinability of critical pairs, CPs for short, and checking of other properties and general correctness of the original TRS specification. For the methodology to be practical, the translation has to take in account several aspects of both ELAN and PVS.

The translation process involves several different components and files. There is one key central component responsible for the translation, but other components are required to make it effective and usable. Figure 1 shows these components and how they interact. Files and software are shown as rectangles and boxes with rounded corners, respectively. They have the following functions and contents:

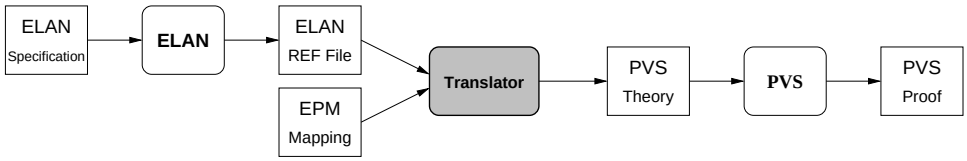


Figure 1: General flow

ELAN Specification: represents the set of files of a given ELAN specification. Most specification will have at least two files: *top level logic description* and *module* files. Some specifications will have additional files called *specification* files. All these files are required to use the ELAN interpreter with that specification.
ELAN: This is the ELAN interpreter. It can be used to run specifications. Its role in the translation process is to export the specification into a ELAN REF file.
ELAN REF File: is used by the ELAN Compiler to generate C code for the given specification. This allows ELAN to create executable programs that run a specification. The content is very suitable for computer analysis, and hold all elements necessary for the generation of the specification in other languages. However the specification does not include any variable names (using something similar to De Bruin's notation for $\lambda-$calculus). This file includes many rules that are not visible in the *module* files. These rules come from imported types and modules, as well as rules generated by properties such as associativity and commutativity.
EPM Mapping: is used to control the translation process and behavior. EPM stands for *ELAN to PVS mapping*. It provides naming for variables, translation for operators and function symbols as well as naming for the rules. ELAN does not have naming for rules, but PVS requires names for declarations such as the ones that will represent the rules. Having meaningful names is important when trying to use rules in proofs. As stated above ELAN will generate internal rules and import rules for modules and types that may be irrelevant to the verification of the specification. To handle this the EPM file can define rules as irrelevant or ignorable.
Translator: is the key component. It will read the ELAN REF file and the EPM mapping files to load the rules into structures that can easily be manipulated. Then, it exports the rules to a PVS file, generate the CPs and provide means to check the CPs (without taking in account the feasibility of their conditions). The translation process will be described in greater detail in section 3.
PVS Theory: PVS files contain the theory declarations. Those generated by the translator will have axioms for the rules of the term rewrite specification, lemmas for the CPs, and function and type declarations, all derived from the original specification.
PVS: represents the PVS runtime that can read theory files, do type checking and allow users to prove the theory's declaration.
PVS Proof: once the user has proved a theory element or declaration, the proof is saved in a proof file. This allows them to be run in batch to re-validate a theory even if some aspects of the theory have changed.

Our translation does not include built-in decision algorithms for deciding whether these conditions are or not satisfiable. Consequently, trivial critical pairs are maintained and their elimination is let for the PVS phase, where one can apply built-in decision procedures of this proof assistant. One may question why the translation does not handle the conditions of the CPs eliminating all CPs with unfeasible conditions, that will trivially proved in PVS. The main reason for not doing so is that PVS has these logical capabilities and introducing decision procedures could produce a very complex translator giving yet another proof engine evaluator which is not the goal in our methodology.

3 TRANSLATION FROM ELAN TO PVS

The translation from ELAN specifications to PVS theories is the cornerstone of our methodology. The key idea is that using a semantically "intelligent" translation one can emulate the rewriting based specifications and use the proof assistant to check both term rewrite properties as well as non rewrite properties such

as logical correctness and integrity. The intelligence comes from the fact that the translation preserves the semantics of the original TRS. To achieve this, the translation mechanism must respect the semantics of the given input specifications and preserve typing information. PVS and ELAN have a somewhat compatible expressive power, but the translation has to deal with differences between them. Within any ELAN specification there are three key elements: **Types** called *sorts* in ELAN; **Function Symbols** called *operators* that represent operator and function symbols and **Rules** that represent the rewrite rules of TRSs. All these elements are listed in the ELAN REF file and must be transported into the PVS theory. Once all of these are loaded, it is possible to generate CPs, which should be included in the PVS theory.

3.1 Types and Function Definitions

ELAN and PVS are strongly typed system. Both also have parametric types (such as list[int] for list of integers). However PVS has a much more powerful typing system, including dependent and recursive types. A naive approach is to translate a type from one system to the other directly *name per name*. So an ELAN *sort* of *mytype* become a PVS type declaration: mytype: **TYPE** .

This approach will generate syntacticly and semantically correct PVS theories, however with complex types such as lists and arrays, this strategy may produce unprovable TCCs. These complex types can be translated into **DATATYPE** types. These type declarations include means to declare constructors and nil elements, and will normally eliminate TCC regarding constructed types.

```
sort mylist; end
operators global
    mynil: mylist;
    @.@: (int mylist) mylist;
end
```

```
mylist: DATATYPE BEGIN
    mynil: mynil?
    o(v:int, l:mylist): mylist_cons?
END mylist
```

Figure 2: Constructed type for lists in ELAN(left) and PVS(right)

As an example, a list can be declared in ELAN as shown on the left side of Figure 2. In the **operators** block each statement defines a function or operator symbol. '@' is a placeholder for a variable of a specific type. The domain of the function is listed within the parenthesis and the range is the name after the parenthesis. So the *mynil* operator defines a constant of type *mylist* and '.' defines and operator of type $int \times mylyst \rightarrow mylist$. The type *mylist* shown above can be translated into the PVS type declaration mylist : **DATATYPE**. The declaration for the list in PVS is shown to the right in Figure 2. The last identifiers after the colon (*mynil*? and *mylist_cons*?) are predicates that can be used elsewhere to determine whether a variable of that **DATATYPE** is constructed by that constructor or not. In ELAN predicates like *mylist_cons*? can be expressed with the rule: [] isMylist_cons(v . l) => **true end**. PVS requires that a predicate be specified for each constructor. These predicates are used in the construction of recursive functions or as rule triggers. More elaborate types may require declaring types manually and turning off type translation for that type.

Another important element of ELAN specifications that must be translated is the type description of the operator and function symbols. This should be done carefully because both ELAN and PVS allow different type descriptions for the same symbol. Since ELAN has a more powerful lexical engine, some operators cannot be translated directly to PVS. For example $r_1 \mapsto Load(r_2)$ can be written in ELAN as r1|−Load(r2) which cannot be translated directly to the language of PVS. To solve this issue, operator symbols that are not supported in PVS must be translated to different symbols. For instance, the ELAN operator @|−Load(@): (int int) instruction can be declared in PVS as Load:[int,int−>instruction]. The general PVS type description is in the form $t_1, t_2, \ldots, t_n \rightarrow i$. Notice that the example above can also be considered a constructor for a **DATATYPE**.

3.2 Rules

ELAN *rules* are translated to PVS *axioms*. This design decision was taken since translating the rules to functional definitions would not necessarily yield adequate functional definitions, and because PVS can emulate TRS rewriting with axioms during proofs. Within the axiom definition the rewriting operator $\rightarrow$ has to be replaced by the equality operator $=$. However, equality in PVS can be used in any direction. Most proof commands that refer to equalities will do so from left to right by default. This is the expected

direction for a TRS. Nevertheless, the use of equality requires special care in the proof to avoid using the equality in the *wrong* direction. That respected, the axioms emulate the rewrite rules correctly in the proof of lemmas.

The emulation of a rewrite step within a proof using the **REWRITE** proof command. This command uses lemmas or axioms and a term or set of terms as arguments, attempting to find unifiers between any sub-term of terms being rewritten and the left side of the equality in the lemma or axiom. If a match is found, it replaces that sub-term by the right side of the axiom or lemma. So $t \xrightarrow{axiom} t'$. Which is similar to the semantic of applying a rule in a TRS, $t \xrightarrow{R} t'$. One caveat of the **REWRITE** command is that it can be used from right to left, although the default is left to right.

A unconditional rule written in ELAN as [] a => b **end**, meaning that a will be rewritten to b. The empty braces [] mean the rule is unlabeled, and tells ELAN that this rule is not used by any specific rewriting-logic strategy. ELAN can use strategies to control the application of rules. Strategies are not dealt by the translation, although this could potentially change the results of the translations. This simple rule is translated to the PVS statement r0: **AXIOM** a=b.

Every axiom and lemma in PVS must have a unique name. These names could be generated automatically, but this hampers usability. Users must apply rules by their names during proofs, thus they should have mnemonic names. Besides that, proofs will have references to the names. If name change, any proof that used the old name will become invalid, forcing the user to redo the proof.

ELAN is a conditional rewriting framework. The ELAN statement [] a -> b **if** c **end** means that a will be rewritten to b if the condition c can be evaluated as true. The statement above can be translates to the PVS term: r1: **AXIOM** c **IMPLIES** a=b. This imitates the way rules work in ELAN, since the axiom can only be applied if PVS can determine that the condition c holds.

ELAN and PVS use *where* constructions to simplify rules and make them more readable. These constructions assign values to variables that appear on the right side of the rules. There are however some differences between the two languages. In ELAN the *where* constructions refer to any variable and condition, in PVS the variable used in conditions or on the left side of a axiom or lemma declaration must be defined before they are used in the statement. This is done with a **LET** ... **IN** expression, which has a similar semantics to a regular *where* construction.

The following example illustrates the way a simple *where* construction is translated. Consider the ELAN rule [] a => c **where** c:=()op(a) **end**. The ELAN assignment operators are in the form :=() or :=[], where the parenthesis or braces can contain the name of a strategy that should be used to evaluate the right side of the assignment. This rule translates to the PVS declaration: r2: **AXIOM** a=c **WHERE** c=op(a). However if the ELAN *where* construction affects a variable in a condition, like in the following rule:

[] a => b **where** c:=()op(a), **if** c

it must be translated slightly differently. Since the condition has to come before the rule's equality, and the variable of the condition has to be defined in the *where* construction, these must be transformed into the **LET** ... **IN** expression such as:

r3: **AXIOM LET** c=op(a) **IN** c **IMPLIES** a = b

Variables must be declared prior to their use in the rules in both languages. In TRSs right side variables have to occur also in left side of the rules or be determined by left-side variables in a *where* construction. However PVS allows variables to be declared and typed per theory declaration (lemma or axiom). This is done with a **FORALL**$(v_1 : t_1, .., v_n : t_n)$: ... construction (where the v_i's are variables and t_i's their types). This is an alternative to the use of global variables and ensures that translation is done correctly since undeclared variables will cause errors during the type checking of the theory. To illustrate how the variable are declared, assuming a of type *int* and c is a *boolean* variables and b is a constant of type *int* declared previously, the previous rule would actually be translated as:

r4: **AXIOM FORALL**(a:int): **LET** c:boolean=op(a) **IN** c **IMPLIES** a = b.

Additionally, the ELAN rule construction block **choose try** ... **end** should be handled. These blocks allow for different assignment to variables depending on conditions provided within the block. Each group started by a **try** represents a set of variable assignments and condition that must be meet. If the condition hold, that set of assignments is accepted and no additional blocks are evaluated. Observe the following block.

```
[] f(x) => g(y,z) choose try where y:=()h(1,x) where z:=()g(y,x) if x < 0
                  try where y:=()h(x,1) where z:=()g(x,y) if x > 0
                  try where y:=()0 where z:=()1 end
```

PVS provides the **COND** or *condition* expression that has a very similar semantics to that of the ELAN **choose try** ... **end** blocks. However, each expression returns a single value of a specific type, and is used to assign a value to a single variable. In ELAN, several variables can be affected by the same condition expression. To handle this, translation must de-multiplex the variables into different PVS **COND** expressions. Assuming all the variables are of type *int*, the rule shown above can be translated to the following PVS declaration.

```
rcond: AXIOM FORALL(x:int): f(x)=g(y,z)
       WHERE
         y=COND x<0 -> h(1,x), x>0 -> h(x,1), ELSE -> 0 ENDCOND,
         z=COND x<0 ->g(y,x), x>0 ->g(x,y), ELSE -> 1 ENDCOND
```

Each condition in the **COND** expression is expressed in the form $c_i \rightarrow v_i$, which means that if c_i holds v_i is the value of the expression. The condition ELSE $\rightarrow v_{else}$ is used if none of the previous conditions is true. Notice that in this example the conditions are the same in both **COND** expressions, just the values returned change. This translation allows PVS to emulate the general semantics of the ELAN **choose try** ... **end** blocks.

The options taken in the translation of rules were carefully analyzed and tested to ensure the preservation of the TRS semantics as well as the usability of the generated PVS theories. Although the translation can produce quite large PVS declarations, a user familiar with the ELAN specifications can easily identify which rule he is observing and determine that the translation matches the original rule. This knowledge is essential during verification, since users will be choosing which axioms (i.e. ELAN rule) to apply as they issued proof commands.

ELAN strategies were not translated to the PVS theory. In ELAN they are used to suggest to the interpreter ways in which the rules should be used. If rules where applied in a non deterministic form, recursion and other looping structures could cause the large consumption of resources that could render computing a result unfeasible. The strategies were ignored (for the time being) because it is assumed that an intelligent user will be making the decision on which rules to apply and this will avoid the problems of random rule selection. This does however limit the methodology to checking just the rules, even though strategies may be an important part of the specification.

3.3 Critical Pairs

Once all the mandatory elements of an ELAN specification have been handled, it is possible to generate CPs. These have no ELAN equivalent, though one could attempt to see how ELAN would handle all of them as queries to the ELAN engine. However this approach is not very practical for testing the joinability.

CPs are generated by overlapping left-hand sides of rules, through the unification of left-hand side of one rule to a non variable sub-term of the left-hand side of the other rule, and then applying one rule to the unified term to get the first element of the pair and the other rule to obtain its second. So lets say the following rules have left-sides such that one non variable sub-term of the one unifies the other:

$$R_1 : l_1 \rightarrow r_1 \quad \text{and} \quad R_2 : l_2 \rightarrow r_2$$

Assuming this unification produces the term u and that: $s \xleftarrow{R_1} u \xrightarrow{R_2} t$, so the pair is $\langle s,\ t\rangle$

CPs are generated to determine whether they are joinable or not, in order to apply rewriting theory to deduce (local-)confluence of the whole TRS. CPs are translated to the PVS theory as conditional equational *lemmas* that need to be proved. The joinability of CPs is determined by proving these lemmas. So for the critical pair $\langle s,\ t\rangle$ one has to prove by rewriting that s and t can be made equal. The corresponding PVS declaration is the lemma: cp1: **LEMMA** s = t.

Many of the rules that create CPs will have conditions associated to them, and will likely have *where* constructions. Both of them must be considered when creating lemmas corresponding to the CPs. These constructions can be joined, assuming they are properly instantiated for variable renaming and unification.

To illustrate this lets take a somewhat complex set of rules (for simplicity the variable are all distinct):

```
[] f(x,h(y)) => g(h(z),h(z)) where z:=()x+y if x>=h(z) end
[] f(s,t) => f(v,z) where v:=()g(s,t) where z:=()g(s,t) if s<t end
```

The left-hand side of both rules can be unified to $f(x, h(y))$ (replacing s by x, and t by $h(y)$). Applying the first rule to the unified term, the following term will appear:

```
g(h(z),h(z)) where z:=()x+y if x>=h(z)
```

And applying the second rule the following term will appear:

f(v,u) **where** v:=()g(x,h(y)) **where** u:=()g(h(y),x) **if** x<h(y)

So the resulting PVS lemma must have the pair's terms and all their condition. The general consideration used for rules must be followed in the CPS. Conditions for rule usage, must be translated into **IMPLIES**. The critical pair from the example rules above should be expressed in PVS as follows:

```
cp2: LEMMA FORALL(x,y:int): LET z=x+y IN x<h(y) AND x>=h(z) IMPLIES
      g(h(z),h(z)) = f(v,u)
      WHERE u=g(h(y),x), v=g(x,h(y))
```

The equality in line 2 is the key test for joinability. The *where* construction from the second rule has to be placed before the main equality because it is needed in the **IMPLIES** clause (first line). Line 3 is the *where* construction from the second rule. On a properly constructed rewrite based specification most of the CPs will have inconsistent conditions, eliminating the CPs altogether. This is a consequence of the fact that the specifier tries to define rewrite rules (for defining each function involved in the specification) whose conditions are exclusive. However, in verifying CPs, even these trivially dispensable, it can appear several problems within the specification as will be discussed in the next session.

4 PRACTICAL EXPERIENCES

This section explores some of the practical experiences in using the methodology and the translator. First we explore the verification of a simple processor, a basic non speculative processor, from the work described in [3]. During this verification several lessons on translation options and actual errors where found in the original specification. Next we show how to extend the verification to operational correctness. This section is an example of use, not a necessary means of specifying the processor or mandatory route to a formal verification.

4.1 Validating an AX Basic Processor Specification

Before introducing the experiences it is important to describe the AX processor. It is a simple RISC processor with 7 instructions: *Load* loads a register with a value from the memory, *Store* saves register contents to memory, *Op* adds two register, *OpE* compares two registers and store the result on another register, *Jz* branches the program if a register is zero, *Loadc* loads a register with a constant value, and *Loadpc* saves the program counter to a register. The *System* has a random access *memory* (implemented as a list of cells) and a *Processor*. The processor is defined as a *program counter*, a *program memory* (which is read only), and a *register file* (list of registers).

During this verification several interesting aspects of the implementation where revealed. These points came from proving both type checking constraints (TCC) and the critical pair joinability. Several of the CPs where unjoinable, one of them revealed a bug in the implementation. All of the issues had limited impact on the actual use of the specification because ELAN evaluates rules in order. However they could cause problems if the order were to be changed or on actual hardware. Some of the difficulties in proving lemmas and obligations derived from options in the translation. Different strategies produced better results.

4.1.1 Bug: Bad Operation Definition

One of the first interesting problems found in the specification had to do with overlapping comparison operations. The $valueofOpE(m, r_1, r_2)$ is used by the *OpE* instruction and is defined as: r_1 and r_2 are registers to be compared, and m is the type of comparison to be made. When $m = 3$, $(\leq)$ the specification had a flaw, as show by the rules below.

```
[] valueofOpE(3,x,y) => 1 if x <= y end
[] valueofOpE(3,x,y) => 0 if x >= y end
```

These rules can be overlapped producing a critical pair, which is unprovable because $x = y \implies 1 = 0$ which is clearly an error. The critical pair generated by the translator is shown below (in PVS).

```
OpE3_lteXOpE3_gte: FORALL(x:int,y:int): x >= y AND x <=y IMPLIES 1=0
```

This error does not produce problem when using the specification in ELAN because the rules are applied in order. However if the order where to be changed the specification would produce erroneous results since when $x = y$ both $valueofOpE(3, x, y) \rightarrow 0$ and $valueofOpE(3, x, y) \rightarrow 1$ are valid reductions.

4.1.2 Register File Critical Pairs

In the AX implementation, register files are implemented as lists of registers. The register is constructed by a term $Reg(i, v)$ where i is the index number of the register and v is it's value. The rules for the update of the register file are shown below.

```
1  [] insertRF(nilr,r,v) => Reg(r,v).nilr end
2  [] insertRF(Reg(i,j).rf,r,v) => Reg(r,v).rf if r == i end
3  [] insertRF(Reg(i,j).rf,r,v) => Reg(i,j).insertRF(rf,r,v) end
```

The first rule is used to add a register to an empty register file or to the end of the file. The second rule is used to replace a register value in the register file, while the third rule advances through the register file. When CPs are analyzed the following critical pair appears from overlapping the second and third rules.

```
irf_foundXirf_adv: LEMMA FORALL(v:int,r:int,rf0:RF,j:int,i:int):
  r=i IMPLIES (Reg(i,j) o insertRF(rf0,r,v))=(Reg(r,v) o rf0)
```

The critical pair has a satisfiable condition and can not be trivially joined. Adding a condition of $r \neq i$ to the third rule changes the conditions of the critical pair turning them unsatisfiable and allowing this way the elimination of the CP. Again the ELAN specification works properly because the rules are executed in order. However, if the second and third rules are swapped the register file specification does not work at all. Adding the condition may seem redundant, but makes the specification clearer and ensures that order will not affect the result of the system.

4.1.3 Instructions Critical Pairs

In the construction of the AX processors a set of operators construct the instructions for the processor. Instructions are constructed using operators like $Jz(r, jaddr)$ and $Load(r, addr)$. The main processing rule has the following format, where Instruction represents one of the AX instructions (for example *isinstJz*):

```
[Instruction] Sys(m, Proc(ia, rf, prog)) =>
    Sys(m, Proc(ia+1, insertRF(rf,r,v), prog))
              where instIa :=() selectinst(prog,ia)
              if isinst<Instruction>(instIa)
              ...
```

The key condition for selecting one of these rules is the $isinst\langle Instruction\rangle$ (like $isinstJz$, and so on). Each of these term takes an instruction to a boolean and indicates if the instruction is constructed by a *Load*, Jz and so on. When two of these rules are checked for CPs, their left-hand sides are identical and will produce a critical pair. There are conditions on each of these CPs which comes down to something like:

```
... isinstJz(instIa) AND isinstLoad(instIa) IMPLIES ....
```

It is obvious that since different constructor create each instruction, no instruction can be both a Jz and *Load* at the same time (or any two instruction constructor for that matter). However PVS does not have that information and the CPs cannot be dismissed or proven joinable. Since $isinst\langle Instruction\rangle$ is not reducible in this form there is no way to get around this problem without some changes in the translation or PVS file.

Several approaches were used to solve this issue. The first approach involve telling PVS that if a instruction is of one type it cannot be of other types. This is done by introducing axioms like:

```
exJz: FORALL(instIa:instruction): isinstJz(instIa) IMPLIES
      NOT isinstLoad(instIa) AND NOT ....
```

This informs PVS that the constructors are different and allow the CPs to be proven unfeasible due to false conditions. Other solutions, which we omit here, involved changing the $isinst\langle Instruction\rangle$ declarations into terms that could be reduced, and that would yield conditions that PVS could determine as unsatisfiable. Such a decoding the instructions into number.

However the most elegant solution to this problem comes from simply changing the type declaration for *instruction* from a simple type to a constructed **DATATYPE**. In addition to this, the predicate for each constructor must match those used in the condition of the rules, that is the $isinst\langle Instruction\rangle$. Thus the *instruction* type in constructed in the following manner (shown in PVS):

```
instruction: DATATYPE BEGIN
    ...
    Load(r:int, addr:int) : isinstLoad
    Jz(r:int, jaddr:int) : isinstJz
    ...
END instruction
```

This translation allows PVS to determine that only one constructor can be used at each time. This makes the CPs formed from the instruction processing rules all trivially unsatisfiable. This is a trivial translation option, but shows how important the translation options can be.

4.1.4 Jz Condition Clause TCC

The rule for the Jz instruction provided another proof challenge. In it's original construction (partially shown below) it has a **choose try .. end** block. As stated in the translation (section 3) this can be converted to a **COND** block. However in doing so this rule and any critical pair that uses it produces a TCC.

```
choose
   try where nia:=()ia+1 if valueofReg(r1,rf) != 0
   try where nia:=()valueofReg(r2,rf) if valueofReg(r1,rf) == 0
```

The reason for this is that PVS requires that all conditions in a **COND** block be pairwise disjoint. Since this block has no *else* condition, proving this TCC requires that $valueofReg(r_1, rf) = 0$ as well as $valueofReg(r_1, rf) \neq 0$ cannot hold at the same time. This is trivially true, but PVS cannot determine this immediately when type checking. One way to solve this is to eliminate the **if** expression from the second **try** block. This forces the translation to use a **ELSE** condition in the **COND** block which eliminates the TCC.

4.2 Simple Example of Operational Correctness

This examples involves a simple ELAN specification with few rules, and then an exercise of the translation and working with the specification. The PVS theory resulting from the translation includes four CPs and is complemented with some lemmas that should be proved to obtain the correctness of the specification. This provides the reader with an example of the potential benefits of using this methodology to verify the ELAN specifications.

The ELAN specification used in this example provides two sets of rules. Rules affecting $agp(s, n, i)$ function, creates a list of elements where each element is the $i-$th elements of an arithmetic progression. The list will have n elements, with initial value of i and a increment of s between elements. Second set of rules are those affecting the $listsum(l)$ and $listsum\_r(l, s)$ term, they take a list l of elements and adds them up, providing the sum s of all the values in the list. The ELAN specification also define the list of integer with the type *ilist*, which is constructed using the $\circ$ operator and the *nililist* for an empty list. The *agp* rules are both conditional, insuring that the list can be assembled even with $n < 0$. The rules are shown below (with the actual line number for each rules).

```
20 // apg ( step, count, increment)
21 [] apg(s,n,i) => nililist if n<=0 end
22 [] apg(s,n,i) => i.apg(s, nc,ni)
23    where nc:=() n−1 where ni:=()i+s if n>0 end
30 [] listsum(nililist) => 0 end
31 [] listsum(l) => listsum_r(l,0) end
32 [] listsum_r(nililist,s) => s end
33 [] listsum_r(h.l,s) => listsum_r(l,ns) where ns:=() h+s end
```

Using proper translation of types, the translation to PVS generated no TCCs and four CPs, all with unsatisfiable conditions. In order to verify the operational correctness, the PVS theory included additionally three lemmas and a function definition. These declarations are used to prove the operation correctness of the specification as follows:

1. $ilist\_length(l)$ is a recursive function that determines the size of a *ilist*. This is a pure PVS definition, as shown below.

```
39      ilist_length(l:ilist): RECURSIVE int = CASES l OF
40              nililist : 0, o(h, t) : 1 + ilist_length(t)
41      ENDCASES MEASURE l BY <<
```

The **MEASURE** ... **BY** is required by PVS to indicate that this recursion is well defined.

2. The *apg_len* lemma states that a list generated by the *apg* rules with argument n has length of n. This is a very simple lemma and seems fairly obvious, never the less it's an example of proving that the rules really do what was intended. It's defined in PVS as follows:

```
45  apg_len: LEMMA FORALL(n:nat,i,s:int): ilist_length(apg(s,n,i))=n
```

3. The *listsum_val_ext* lemma is used in the proof of the next lemma. It was required because the way the *listsum* is defined in the rules, storing the sum s within the functional symbol. (See line 32 of the ELAN specification).

```
47  listsum_val_ext: LEMMA FORALL(v,c:int, l:ilist): listsum_r(v o l,c)=listsum(l)+v+c
```

4. The last lemma *listsum_val* shows that the $listsum(agp(s,n,i))$ with s the increment, i the initial value and n has the values of the closed form of the sum of elements in an arithmetic progression (where a_i is the $i-$th element of the arithmetic progression): $((a_0 + a_n) \times n) \div 2$. In PVS this is expressed as follows:

```
51  listsum_val: LEMMA FORALL(n:nat,i,s:int): listsum(apg(s,n,i))= ((2 * i + s * l −s)*l)/2
52      WHERE l:int=ilist_length(apg(s,n,i))
```

The proof of the first lemma can be done with a simple induction on n. However the ELAN specification specifies n as $n \in \mathbb{Z}$, which is not suitable for an induction. The solution is to change n to a $n \in \mathbb{N}$ in the definition of the lemma. Since *agp* is defined for $\mathbb{Z}$ and $\mathbb{N} \in \mathbb{Z}$, the definition will work properly.

The next lemma, *listsum_val_ext*, is also proved by an induction, but this time it is done on the l (an *ilist*). This is an interesting feature of PVS, where any well-founded type can be used for an induction. This lemma is required for the next lemma since the sum of the list is store within the $listsum\_r(l,s)$ function as the s value. However for the next induction it is important to have the value of the accumulated sum outside of the function symbols. This is what this lemma proves, that on each step of the *listsum*:

$$\forall v, c \in \mathbb{Z}, l \in ilist : listsum\_r(v \circ l, c) = listsum(l) + c + v$$

The last lemma *listsum_val*, is proved by induction using the previous lemmas to assist in the proof. Again in order to have an induction it is necessary to restrict n to $n \in \mathbb{N}$. As shown above using PVS we can check more than the basic properties of a rewrite logic specification. One other interesting aspect of using PVS is that the ELAN specifications goes through a comprehensive type checking.

5 CONCLUSIONS AND FUTURE WORK

The proposed methodology is easily applicable for general TRS based specifications and generating the target lemmas corresponding to (joinability of) CPs and using the power of a proof assistant such as PVS simplifies the verification of operational correctness. And the most important, it allows for dynamic and quick execution of changes in the specification without losing work in proving its correctness, since PVS can rerun proofs, assuming changes did not invalidate them. PVS proved to be very suitable for this task since it can emulate the rewriting semantics involved in conditional rewriting rules and conditional critical pairs very well, and may be used to prove other properties, such as operational correctness.

The experience with the basic AX specification showed various important options concerning the translation process. The initial naive approach produced many unprovable TCCs and had some critical pairs that could not be discarded. With different translation strategies and minor adjustments to the specification, it was possible to create a PVS theory with no TCCs and whose CPs where all discarded. Even with all the care taken in the translation the fact that PVS does not have an order when applying lemmas and axioms, and that types can be handled in different forms can play an important role in the number of TCCs and the complexity of proving properties. Larger specifications are being converted and tested after the initial success of the methodology. Future work includes creating proof strategies to assist proving the CPs between PVS, and making the translation aware of the ELAN strategies.

Finally, we believe the proposed methodology is a step forward in the direction of describing a verification mechanism for innovative new hardware technologies such as reconfigurable systems. Current work in this setting point out the translation of rewriting based specifications of algebraic operations to alternative equivalent expressions via different logical strategies in ELAN, which are then translated to the language of the ALE-X compiler of the Xputer reconfigurable architecture and subsequently mapped into the PACT XPP commercial architectures [13].

References

[1] Arvind and X. Shen. Using term rewriting systems to design and verify processors. *IEEE Micro Special Issue on "Modeling and Validation of Microprocessors"*, 19(3):36–46, May/June 1999.

[2] M. Ayala-Rincón, R. P. Jacobi, L. G. A. Carvalho, C. Llanos, and R. Hartenstein. Modeling and Prototyping Dynamically Reconfigurable Systems for Efficient Computation of Dynamic Programming Methods. In *17th Symp. on Integrated Circuits and System Design*, pages 248–253. ACM Press, 2004. Journal version to appear in *Genetics and Molecular Research, 2005.*

[3] M. Ayala-Rincón, R. Maya Neto, R. P. Jacobi, C. Llanos, and R. Hartenstein. Applying ELAN strategies in simulation processors over simple architectures. In *2nd Int.Workshop on Reduction Strategies in Rewriting And Programming*, volume 70(6) of *ENTCS*, pages 127–141. Elsevier, 2002.

[4] M. Ayala-Rincón, R. B. Nogueira, R. P. Jacobi, C. Llanos, and R. Hartenstein. Modeling a reconfigurable system for computing the FFT in place via rewriting-logic. In *16th Symp. on Integrated Circuits and System Design*, pages 205–210. IEEE CS, 2003.

[5] F. Baader and T. Nipkow. *Term Rewriting and All That.* CUP, 1998.

[6] J. Becker and R. W. Hartenstein. Configware and morphware going mainstream. *Journal of Systems Architecture*, 49:127–142, 2003.

[7] D. Cyrluk, S. Rajan, N. Shankar, and M.K. Srivas. Effective theorem proving for hardware verification. In *Theorem Provers in Circuit Design*, volume 901 of *LNCS*, pages 203–222. Springer, 1994.

[8] J. C. Hoe and Arvind. Open-Centric Hardware Description and Synthesis. *IEEE T. on CAD of Integrated Circuits and Systems*, 23(9):1277–1288, 2004.

[9] D. Kapur and M. Subramaniam. Mechanizing Verification of Arithmetic Circuits: SRT Division. In *Proc. 17th FSTTCS*, volume 1346 of *LNCS*, pages 103–122. Springer, 1997.

[10] D. Kapur and M. Subramaniam. Using an Induction Prover for Verifying Arithmetic Circuits. *J. of Software Tools for Technology Transfer*, 3(1):32–65, 2000.

[11] N. Martí-Oliet and J. Meseguer. Special Issue on Rewriting Logic and its Applications. *TCS*, 285(2), 2002.

[12] S. P. Miller and M. Srivas. Formal verification of the AAMP5 microprocessor: A case study in the industrial use of formal methods. In *Workshop on Industrial-Strength Formal Specification Techniques*, pages 2–16. IEEE CS, 1995.

[13] C. Morra, J. Becker, M. Ayala-Rincón, and R. W. Hartenstein. FELIX: Using Rewriting-Logic for Generating Functionally Equivalent Implementations. In *15th Int. Conference on Field Programmable Logic and Applications - FPL 2005.* IEEE CS, 2005. To appear.

[14] E. Ohlebusch. *Advanced Topics in Term Rewriting.* Springer, 2002.

[15] S. Owre, J. Rushby, N. Shankar, and D. Stringer-Calvert. PVS: An Experience Report. In *Applied Formal Methods-FM-Trends 98*, volume 1641 of *LNCS*, pages 338–345. Springer, 1998.

[16] S. Owre, J. M. Rushby, N. Shankar, and M. K. Srivas. A tutorial on using PVS for hardware verification. In *Theorem Provers in Circuit Design (TPCD '94)*, volume 901 of *LNCS*, pages 258–279. Springer, 1994.

[17] R. Hosabettu and G. Gopalakrishnan and M. Srivas. Formal verification of a complex pipelined processor. *Formal Methods in System Design*, 23:171–213, 2003.

[18] J.M. Rushby, S. Owre, and N. Shankar. Subtypes for Specifications: Predicate Subtyping in PVS. *IEEE T. on Software Engineering*, 24(9):709–720, 1998.

[19] C. J. van Rijsbergen, editor. *Term Rewriting Systems.* CUP, 2003.

Prueba de propiedades en la caché de un servidor funcional de video bajo demanda*

J. Santiago Jorge, Víctor M. Gulías, Alberto Valderruten, David Cabrero
Universidade da Coruña, Dept. de Computación, MADS - LFCIA
Campus de Elviña, s/n., 15071 A Coruña, España
sjorge@udc.es, gulias@dc.fi.udc.es, valderruten@udc.es, cabrero@udc.es

Abstract

This paper describes the formal verification of part of a real-world application: a distributed *Video-on-Demand* server, implemented with the concurrent functional programming language Erlang. As the application of a formal method to a large system is extremely difficult or even impossible, we verify relevant properties of a part of this system separately, so that when later we combine other partial results, conclusions on the whole system could be drawn. The development consists on two steps: first, the definition in the Coq proof assistant of the algorithm and some relevant properties to be proved; second, the codification of the theorems with the help of some new tactics resulting from the abstraction of verification patterns common to different proofs.

Keywords: Formal methods, software verification, theorem provers, functional programming, real-world applications

Resumen

Este artículo describe la verificación formal de una parte de una aplicación del mundo real: la implementación de un servidor distribuido de vídeo bajo demanda, que ha sido desarrollado en el lenguaje funcional concurrente Erlang. Como la aplicación de un método formal de verificación al sistema completo resulta extremadamente difícil o imposible debido a su tamaño, verificamos propiedades relevantes de una parte del mismo de forma separada, de modo que luego, combinando otros resultados parciales, se pueda llegar a conclusiones sobre el sistema al completo. El desarrollo consta de dos pasos: primero, la definición en el asistente de pruebas Coq del algoritmo y ciertas propiedades relevantes a probar; y después, la codificación de los teoremas con la ayuda de algunas nuevas tácticas resultantes de la abstracción de patrones de verificación comunes a diferentes pruebas.

Palabras claves: Métodos formales, verificación del software, probadores de teoremas, programación funcional, aplicaciones del mundo real.

1. Introducción

Uno de los costes más altos que aparece en la producción de software es el de la corrección de errores que son detectados con posterioridad al desarrollo del sistema [12, 18]. Aunque la comprobación de la corrección de un programa es una de las actividades más importantes en el desarrollo del software, al mismo tiempo se trata de una labor muy compleja. El método más utilizado para la validación del software consiste en la simulación sobre casos de prueba representativos. Sin embargo, este método no garantiza la corrección del programa analizado, por ser los casos de prueba incompletos en la mayoría de las ocasiones [8]. De modo que incluso código que ha sido ya comprobado puede contener todavía errores.

Este trabajo propone la verificación formal de una parte de la funcionalidad de una aplicación del mundo real: la implementación de un servidor distribuido de vídeo bajo demanda [10] desarrollado con el lenguaje funcional concurrente Erlang [1]. Los *lenguajes funcionales* [4, 6, 13, 16] se presentan como un escenario

*Parcialmente financiado por MCyT TIC2002-02859 y Xunta de Galicia PGIDIT03PXIC10502PN

atractivo a la hora de desarrollar programas que puedan ser analizados formalmente para garantizar su corrección. Esto se debe a la *transparencia referencial*, que describe el estilo de programación en el cual *iguales son reemplazables por iguales*, y nos permite emplear sobre los programas herramientas como el *razonamiento ecuacional* o la *inducción*.

Como la aplicación de un método formal de verificación al sistema completo resulta muy difícil, o incluso imposible debido a su tamaño, aplicamos técnicas presentadas en [14] a la verificación de una parte del mismo de forma independiente, de modo que luego, combinando otros resultados parciales, se pueda llegar a conclusiones sobre el sistema al completo.

La certificación se lleva a cabo en el asistente de pruebas Coq [3, 7] para garantizar su corrección. Los probadores de teoremas proporcionan control y asistencia durante los procesos de verificación; por tanto evitan errores que podrían introducirse en una prueba hecha a mano. El sistema Coq es una implementación del *cálculo de construcciones inductivas* [5, 15], desarrollado en el INRIA. Este sistema es un probador automático de teoremas dirigido a objetivos por medio de tácticas en el estilo del LCF [9].

La estructura del artículo es como sigue: Primero, se describe brevemente el proyecto VoDKA (*Video-on-Demand Kernel Architecture*, `http://vodka.lfcia.org`), presentando las principales características de un servidor de vídeo bajo demanda, así como las decisiones de diseño adoptadas en el proyecto. La sección 3 contextualiza la porción de software objeto de estudio. La sección 4 presenta el modelo Coq usado para describir el algoritmo. La sección 5 muestra algunas tácticas nuevas resultantes de la identificación de patrones comunes a diferentes pruebas. En las secciones 6 y 7 se certifica la corrección del programa. Por último, se presentan las conclusiones.

2. El proyecto VoDKA

En el año 2000 comenzó un proyecto auspiciado por *R*, el operador de cable de la comunidad gallega, con el objetivo de proveer servicios de vídeo bajo demanda a sus clientes

Un servidor de vídeo bajo demanda es un sistema que proporciona servicio de vídeo (u otros servicios multimedia) a muchos clientes que realizan solicitudes en cualquier momento, sin limitaciones temporales preestablecidas. Tiene que satisfacer algunos requerimientos críticos: gran capacidad de almacenamiento, manejo de miles de transacciones concurrentes garantizando tiempos de respuesta razonables e idealmente predecibles, gran ancho de banda, fiabilidad, escalabilidad, adaptabilidad a la topología subyacente, y bajo coste.

Tras analizar los productos de vídeo bajo demanda existentes, se llegó a la conclusión de que la mayoría representaban soluciones no adaptables, no escalables, cerradas y muy caras. Por tanto el objetivo del proyecto fue la construcción de un servidor escalable, tolerante a fallos, multiprotocolo y adaptable (tanto a la topología de la red como a los protocolos de los usuarios finales): el servidor VoDKA.

Los clusters de ordenadores de bajo coste representan una solución asequible para una amplia variedad de aplicaciones que demandan gran cantidad de recursos. Sin embargo, el problema principal era cómo un pequeño grupo de desarrolladores programarían un sistema distribuido tan complejo. Además, como en un primer momento muchos requerimientos no eran conocidos en todo su detalle, el sistema debía de ser tan flexible como fuese posible para adaptarse a nuevos escenarios. Se trataba de un reto adecuado para la programación funcional.

El lenguaje funcional elegido fue Erlang [1], que es un lenguaje de programación funcional concurrente y distribuido, desarrollado por Ericsson para aplicaciones de telecomunicaciones. Erlang es uno de los pocos lenguajes funcionales usados con éxito en aplicaciones del mundo real [19]. Se prefirió Erlang frente a otros lenguajes como Caml o Haskell porque: (a) Tiene construcciones para la computación distribuida y concurrente; (b) Dispone de una amplia biblioteca de herramientas que facilitan el desarrollo; (c) tiene una buena interfaz con lenguajes de bajo nivel (necesarios por razones de rendimiento o para interactuar con dispositivos); (d) resulta sorprendentemente eficiente en aplicaciones distribuidos con miles de procesos concurrentes.

Sin embargo, Erlang presenta algunas carencias: (a) El sistema de tipado: simples errores de tipado no son detectados en tiempo de compilación debido a que no tiene tipado estático; (b) El sistema de módulos: relacionado con el punto anterior, no es posible comprobar que un módulo satisface una interfaz dada. Además el espacio de nombres en los módulos es muy primitivo (plano) aunque existen ahora propuestas para usar un sistema de módulos jerárquico. (c) Cobertura: como la mayoría de lenguajes funcionales, necesita más programadores, más libros, más patrones, más experiencias, etc.

El primer enfoque del diseño se realizó para definir un sistema de almacenamiento distribuido y jerárquico (figura 1). En esta propuesta inicial la jerarquía se compone por niveles especializados descritos de abajo a arriba por:

Nivel terciario (repositorio o nivel de almacenamiento masivo): su objetivo es el almacenamiento de todos los objetos multimedia del servidor. Su principal rasgo es la gran capacidad.

Nivel secundario (nivel de *caché*): El objeto multimedia leído del nivel terciario es almacenado aquí antes de su envío por el nivel primario al cliente. Una técnica de planificación adecuada decidirá qué vídeos se mantienen en la caché y durante cuánto tiempo para satisfacer futuras solicitudes y evitar el acceso al nivel terciario. Un alto *throughput* (es decir, la cantidad de trabajo realizable en un período de tiempo dado) es su característica principal.

Nivel primario (nivel de *streaming*): se encarga de la adaptación de protocolos y del envío de los objetos multimedia en el formato apropiado al cliente.

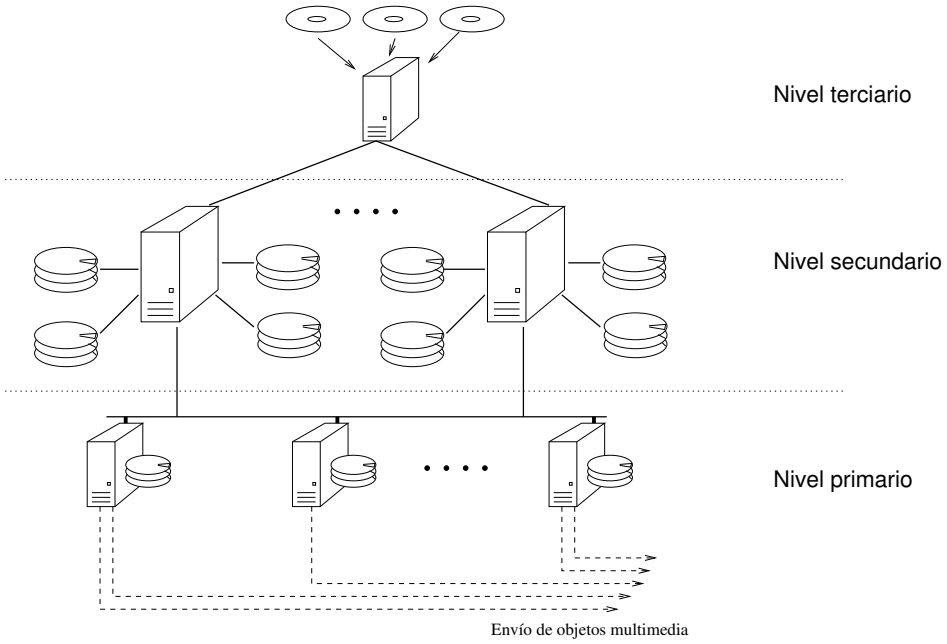


Figura 1: Estructura jerárquica inicial del servidor distribuido de vídeo bajo demanda

Las ideas iniciales del diseño han tenido que ser refinadas para satisfacer las necesidades de producción del sistema. En particular, se necesitó una generalización del diseño porque la arquitectura jerárquica de tres niveles puede ser demasiado compleja para una instalación pequeña, o demasiado rígida para una topología de red compleja, como una red que interconecte redes metropolitanas de cable.

3. El subsistema de caché del servidor funcional de vídeo bajo demanda

Aquí vamos a contextualizar la porción de software que va a ser objeto de verificación. Se han aplicado también métodos formales a otras partes [2, 11].

El disco local de cada nodo del nivel secundario actúa como *caché*; además hay una caché general (se trata de un componente lógico) en el nodo frontal. Cada caché local está dividida en bloques de tamaño fijo (8 MB, por ejemplo) que están numerados. Cada uno de estos bloques son unidades de información.

Cuando llega una petición realizamos la búsqueda en las caches locales, o delegamos la petición del fichero al nivel terciario.

En todo momento tenemos que controlar el espacio ocupado, de manera que cuando realizamos la petición de un objeto multimedia al nivel terciario, de algún modo habrá que reservar el espacio (garantizándolo)

en la caché local de algún nodo del nivel secundario. Si para ello hubiese que liberar unos bloques, hay que garantizar que no estén siendo usados. Para ello cada bloque tiene marcado que está siendo usado por algún usuario (en realidad, el número de peticiones pendientes). Resulta además importante que los bloques de un mismo fichero estén lo más cerca posible, luego las asociaciones de vectores en caché serán lo más secuenciales posibles. Por ello hablaremos de *intervalos de bloques*.

Cada intervalo se representa como un triple (a, b, x) denotando que el intervalo comprendido entre los bloques a y b, ambos incluidos, tiene x clientes pendientes. Guardaremos por tanto la secuencia de intervalos de bloques sobre los que existen peticiones, como una lista de intervalos como sigue:

$$[(a_1, b_1, x_1), \ldots, (a_i, b_i, x_i), (a_{i+1}, b_{i+1}, x_{i+1}), \ldots, (a_n, b_n, x_n)]$$

Por motivo de eficiencia de las operaciones, mantenemos los intervalos ordenados $\forall i, a_i \leq b_i \wedge b_i < a_{i+1}$. Y para ahorrar espacio, mantenemos la secuencia "compactada", es decir: $\forall i, x_i \neq 0$ y $(x_i \neq x_{i+1}) \vee ((x_i = x_{i+1}) \wedge (b_i + 1 < a_{i+1}))$

La función `add` añade una petición sobre un intervalo de bloques a la secuencia de intervalos, devolviéndonos un par con la nueva secuencia de intervalos y aquellos intervalos de bloques que no tenían peticiones antes de la operación. Tras la validación del software mediante la simulación sobre casos de prueba representativos, que permitieron la detección de algunos errores, llegamos al código de la figura 2. Sin embargo la corrección del programa analizado no está garantizado ya que los casos de prueba podrían haber sido incompletos.

Para certificar la corrección del programa, lo reescribimos en otro equivalente en el sistema de pruebas Coq. Cambiamos algún detalle del programa, simplificándolo: la función `add` original usa recursividad terminal mientras que nuestro modelo en Coq es una versión no terminal, y añade una petición sobre un intervalo de bloques mientras que el modelo Coq únicamente añade una petición sobre un bloque. El inconveniente de este método es que no estamos verificando el sistema real sino un modelo del mismo; sin embargo, este proceso incrementa la fiabilidad del sistema [17].

4. El modelo Coq del algoritmo

En esta sección presentamos el código Coq usado en la descripción del algoritmo. En primer lugar definimos los *intervalos* de bloques. Como los bloques están numerados, cada intervalo está representado por un triplete de naturales: las posiciones de comienzo y fin (de los bloques de memoria) y el número de peticiones existentes sobre ellos.

```
Inductive interval: Set:= tuple: nat -> nat -> nat -> interval.
```

Se trata de una definición inductiva con un constructor, `tuple`, que recibe tres números naturales para crear un intervalo.

Una *secuencia* de intervalos es una lista, que representa todos los bloques de memoria con sus peticiones. También concretamos esta definición mediante un tipo inductivo.

```
Inductive seq: Set:= Nil: seq | Cons: interval -> seq -> seq.
```

Los constructores son `Nil` y `Cons`. El primero representa la secuencia vacía, y el segundo toma un intervalo y una secuencia para formar otra secuencia.

Tal como explicamos en la sección anterior, en todo momento mantendremos por motivos de eficiencia la secuencia de intervalos ordenada ascendentemente; es decir, para toda secuencia $[(a_1, b_1, x_1), \ldots, (a_n, b_n, x_n)]$, se cumplirá que $\forall i, a_i \leq b_i \wedge b_i < a_{i+1}$.

La función `add` (figura 3) añade una petición más sobre un bloque de memoria. La definición se realiza por recursión estructural en el segundo argumento: la secuencia de intervalos. Vamos a tener siete alternativas posibles. El motivo de la especial complicación de la definición de `add` es el mantenimiento de la secuencia en su orden ascendente, lo cual implica que en algunos casos habrá que partir un intervalo.

De cara a demostrar la corrección de `add` necesitamos una función auxiliar `nth` que nos devuelva el número de peticiones existentes sobre el n-ésimo bloque de memoria; si el índice estuviese fuera de rango devolvería un valor por defecto, que recibe como argumento. Esta función auxiliar no pertenece realmente al algoritmo pero la necesitamos para formular su especificación. Tendríamos que garantizar su corrección aunque esta tarea se sale del objetivo del presente trabajo, y podría tratarse en una biblioteca de funciones certificadas como las presentadas en [14].

```
add({A,B},Sequence) -> add({A,B},[],Sequence,[]).

add({A,B},RevHead,[{C,D,N}|Tail],Added) ->
  if  B < C ->
        {packrev([{C,D,N},{A,B,1}|RevHead]) ++ Tail,
         lists:reverse([{A,B}|Added])};
      A < C,C =< B,B < D ->
        {packrev([{B+1,D,N},{C,B,N+1},{A,C-1,1}|RevHead]) ++ Tail,
         lists:reverse([{A,C-1}|Added])};
      A < C,B == D ->
        {packrev(lists:reverse(Tail,[{C,D,N+1},{A,C-1,1}|RevHead])),
         lists:reverse([{A,C-1}|Added])};
      A < C,D < B->
        add({D+1,B},[{C,D,N+1},{A,C-1,1}|RevHead],
             Tail,[{A,C-1}|Added]);
      A == C,B < D ->
        {packrev([{B+1,D,N},{A,B,N+1}|RevHead]) ++ Tail,
        lists:reverse(Added)};
      A == C,B == D ->
        {packrev(lists:reverse(Tail,[{A,B,N+1}|RevHead])),
         lists:reverse(Added)};
      A == C,D < B -> add({D+1,B},[{C,D,N+1}|RevHead],Tail,Added);
      C < A,B == D ->
        {packrev(lists:reverse(Tail,[{A,D,N+1},{C,A-1,N}|RevHead])),
         lists:reverse(Added)};
      C < A,B < D ->
        {packrev(RevHead) ++
         [{C,A - 1,N},{A,B,N + 1},{B + 1,D,N}|Tail],[]};
      C < A,A =< D,D < B ->
        add({D+1,B},[{A,D,N+1},{C,A-1,N}|RevHead],Tail,Added);
      D < A -> add({A,B},[{C,D,N}|RevHead],Tail,Added)
  end;
add({A,B},RevHead,[{C,D}|Tail],Added) ->
  add({A,B},RevHead,[{C,D,1}|Tail],Added);
add({A,B},RevHead,[],Added) ->
  {packrev([{A,B,1}|RevHead]),lists:reverse([{A,B}|Added])}.

packrev(L) -> packrev([],L).

packrev(L,[{C,D,Y},{A,B,X}|Tail]) ->
  if (B + 1) == C, X == Y -> packrev(L,[{A,D,X}|Tail]);
     true -> packrev([{C,D,Y}|L],[{A,B,X}|Tail])
  end;
packrev(L,[T]) -> [T|L];
packrev(L,[]) -> L.
```

Figura 2: Programa ERLANG cuya corrección queremos certificar

5. Simplificación del proceso de verificación con nuevas tácticas

Como parte del proceso de verificación, han sido definidas nuevas tácticas *ad hoc*. Son el resultado de la abstracción de patrones de verificación comunes a diferentes pruebas y nos van a permitir eliminar algunos pasos repetitivos cuando varios lemas o algunas partes de los mismos se prueban de forma similar, pero no exactamente de la misma manera.

El sistema de pruebas COQ nos provee de tácticas predefinidas para su aplicación en el desarrollo de las pruebas, y estas tácticas son extensibles por el usuario combinándolas por secuenciamiento o selección, representados por los operadores "`;`" y `Orelse` respectivamente. Además, el sistema permite definiciones parametrizadas de combinaciones de tácticas. Una vez definidas, estas tácticas, proporcionando la lista de términos correspondiente a sus argumentos, pueden ser usadas como cualquier otra táctica.

Por ejemplo, la táctica `Casec` es una *macro* que realiza un análisis de casos sobre un término `H`, que recibe como argumento. Además, `H` es eliminado del conjunto de las hipótesis.

```
Tactic Definition Casec H:= Case H; Clear H.
```

La táctica `IntroInt` descompone un intervalo en el triplete de números naturales que lo forman. Recibe

```
Fixpoint add [n:nat; l: seq]: seq :=
 Cases l of
 | Nil => (Cons (tuple n n (S O)) Nil)
 | (Cons (tuple a b x) l') =>
   Cases (le_lt_dec a n) of
   | (left _) =>
     Cases (le_lt_dec n b) of
     | (left _) =>
       Cases (eq_nat_dec a n) of
       | (left _) =>
         Cases (eq_nat_dec n b) of
         | (left _) => (Cons (tuple a b (S x)) l')                                  (*a=n, n=b*)
         | (right _) => (Cons (tuple a a (S x)) (Cons (tuple (S a) b x) l'))        (*a=n, n<b*)
         end
       | (right _) =>
         Cases (eq_nat_dec n b) of
         | (left _) => (Cons (tuple a (pred b) x) (Cons (tuple b b (S x)) l'))      (*a<n, n=b*)
         | (right _) =>                                                            (*a<n, n<b*)
             (Cons (tuple a (pred n) x) (Cons (tuple n n (S x)) (Cons (tuple (S n) b x) l')))
         end
       end
     | (right _) => (Cons (tuple a b x) (add n l'))                              (*b<n*)
     end
   | (right _) => (Cons (tuple n n (S O)) l)                                     (*n<a*)
   end
 end.
```

Figura 3: Definición de `add` in el sistema Coq

como argumento los nombres que van a recibir los tres números naturales. Aquí hacemos uso de la táctica `Casec`, definida con anterioridad.

```
Tactic Definition IntroInt a b x:= Intro intrvl; Casec intrvl; Intros a b x.
```

Con la expresión `Match Context With` exploramos las hipótesis del contexto de la prueba además de la conclusión por medio de reconocimiento de patrones. Se realiza un análisis de casos, y cuando se cumple algún patrón se aplican las tácticas correspondientes sobre la meta de la prueba sustituyendo previamente las meta-variables por los nombres de las hipótesis correspondientes. La siguiente táctica trata la resolución automática de conclusiones cuando en el contexto haya hipótesis aritméticas contradictorias.

```
Tactic Definition Contradict:=
Match Context With
 | [_:(le ?1 ?2); _:(le ?2 ?3); _:(lt ?3 ?1) |-?] ->
     Absurd (lt ?3 ?1); [Apply le_not_lt; Apply le_trans with ?2; Auto | Auto]
 | [_:(lt ?1 ?2); _:(lt ?2 ?3); _:(le ?3 ?1) |-?] ->
     Absurd (le ?3 ?1); [Apply lt_not_le; Apply lt_trans with ?2; Auto | Auto]
 | [_:(lt ?1 ?2); _:(le ?2 (pred ?1)) |-?] ->
     Absurd (lt (pred ?1) ?2);
       [Apply le_not_lt | Apply le_lt_trans with ?1]; Auto with arith
 | [_:(lt ?1 ?2); _:(lt ?2 (S ?1)) |-?] ->
     Absurd (lt ?2 (S ?1)); [Apply le_not_lt; Auto with arith | Auto]
 | [_:(lt ?1 ?2); _:(lt ?2 ?1) |-?] ->
     Absurd (lt ?2 ?1); [Apply le_not_lt; Apply lt_le_weak; Auto | Auto]
 | [_:(lt ?1 ?1) |-?] ->
     Absurd (lt ?1 ?1); [Apply le_not_lt; Auto | Auto]
 | [_:(le ?1 ?2); _:(le ?2 ?1); _:~(?1=?2) |-?] ->
     Absurd ?1=?2; [Auto | Apply le_antisym; Auto]
 | [_:(lt ?1 ?2); _:(lt ?2 ?3); H:(S ?1)=?3 |-?] ->
     Absurd (lt ?2 ?3); [Apply le_not_lt; Rewrite <- H; Apply lt_le_S; Auto | Auto]
```

Usamos a continuación la táctica recién definida para implementar otra macro. Cuando en la conclusión tenemos una alternativa basada en el resultado computacional `(le_lt_dec a b)`, con `a` y `b` dos naturales cualesquiera, y por las hipótesis del contexto sabemos que una de las alternativas nos lleva a una contradicción, podremos aplicar la táctica `DecLeLt` sobre los argumentos `a` y `b`, que nos librará automáticamente

de una de las dos ramas. Aquí, además de secuenciamiento tenemos un ejemplo de combinación de tácticas mediante selección.

```
Tactic Definition DecLeLt a b:=
  Case (le_lt_dec a b); Auto Orelse (Intros; Try Contradict).
```

Como se observa, las tácticas definidas hasta aquí no son demasiado sofisticadas y sirven principalmente para eliminar algunos pasos repetitivos en el desarrollo de las pruebas.

6. Verificación del modelo Coq

Estudiamos aquí la corrección del algoritmo que añade una nueva petición sobre un bloque de memoria. En cada momento, sobre un bloque de memoria hay un número de peticiones determinado. La propiedad que la función `add` tiene que cumplir es que tras su ejecución, el número de peticiones sobre el bloque al que se le añadió otra nueva habrá sido incrementado mientras que todos los demás bloques permanecerán con el mismo número de peticiones que tenían antes de esta operación:

$$\texttt{nth n (add i l) 0} = \begin{cases} \texttt{nth n l 0} & \text{if } \texttt{i} \neq \texttt{n} \\ \texttt{nth n l 0} + 1 & \text{if } \texttt{i} = \texttt{n} \end{cases}$$

Subdividimos esta propiedad en dos teoremas según `i` sea igual a `n` o no. Mediante la demostración de ambos, certificaremos la corrección de `add` con respecto a su especificación.

```
Theorem nth_add_1: (l: seq; n, i: nat) i=n -> (nth n (add i l) 0) = (S (nth n l 0)).
```

Tanto la definición de `add` como la de `nth` se realizan por recursión estructural en la secuencia de intervalos de bloques, luego procedemos por inducción en `l`.

```
Induction l.
Intros; Rewrite H; Simpl.
```

```
2 subgoals
  ⋮
  H : i=n
  ============================
   (Cases (le_lt_dec n n) of
      (left _) =>
       Cases (le_lt_dec n n) of
         (left _) => (1)
       | (right _) => (0)
       end
    | (right _) => (0)
    end)=(1)

subgoal 2 is:  ...
```

El análisis de casos aplicado sobre `le_lt_dec n n` devuelve o bien el valor `(left _)` con una prueba de `le n n`, o el valor `(right _)` con una prueba de `lt n n`. Como todo número natural es igual a si mismo, la segunda alternativa nos llevaría a una hipótesis errónea $n < n$ y se resolvería de forma inmediata.

Resolvemos por tanto el caso base usando la táctica `DecLeLt n n` definida por nosotros que nos resuelve esta sub-meta; y continuamos con el paso inductivo.

```
DecLeLt n n.
```

```
1 subgoal

  l : seq
  ============================
   (i:interval; s:seq)
    ((n,i:nat)i=n->(nth n (add i s) (0))=(S (nth n s (0))))
    ->(n,i0:nat)i0=n->(nth n (add i0 (Cons i s)) (0))=(S (nth n (Cons i s) (0)))
```

En el paso inductivo comenzamos introduciendo al entorno de las hipótesis los cuantificadores universales, la hipótesis de inducción y la hipótesis del teorema, reescribimos esta hipótesis y realizamos todas las β-reducciones posibles.

```
Clear l;IntroInt a b x;Intros l indH n j Heq.
Rewritec Heq; Simpl.
```

```
1 subgoal
  ⋮
  indH : (n,i:nat)i=n->(nth n (add i l) (0))=(S (nth n l (0)))
  n : nat
  j : nat
  ============================
   (nth n
     Cases (le_lt_dec a n) of
       (left _) =>
        Cases (le_lt_dec n b) of
          (left _) =>
           Cases (eq_nat_dec a n) of
             (left _) => Cases (eq_nat_dec n b) of
                           (left _) => (Cons (tuple a b (S x)) l)
                         | (right _) => (Cons (tuple a a (S x)) (Cons (tuple (S a) b x) l))
                         end
           | (right _) =>
              Cases (eq_nat_dec n b) of
                (left _) => (Cons (tuple a (pred b) x) (Cons (tuple b b (S x)) l))
              | (right _) =>
                 (Cons (tuple a (pred n) x) (Cons (tuple n n (S x)) (Cons (tuple (S n) b x) l)))
              end
           end
        | (right _) => (Cons (tuple a b x) (add n l))
        end
     | (right _) => (Cons (tuple n n (1)) (Cons (tuple a b x) l))
     end (0))
    =(S Cases (le_lt_dec a n) of
          (left _) =>
           Cases (le_lt_dec n b) of
             (left _) => x
           | (right _) => (nth n l (0))
           end
        | (right _) => (0)
        end)
```

Siguiendo la estructura de la función `add` efectuamos los siguientes tratamientos por casos.

```
CaseEq '(le_lt_dec a n) Hle1 le1.
CaseEq '(le_lt_dec n b) Hle2 le2.
Case (eq_nat_dec a n); Intro Heq1. Rewritec Heq1.
Case (eq_nat_dec n b); Intro Heq2.
```

```
5 subgoals
  ⋮
  indH : (n,i:nat)i=n->(nth n (add i l) (0))=(S (nth n l (0)))
  n : nat
  j : nat
  Hle1 : (le a n)
  le1 : (le_lt_dec a n)=(left (le a n) (lt n a) Hle1)
  Hle2 : (le n b)
  le2 : (le_lt_dec n b)=(left (le n b) (lt b n) Hle2)
  Heq2 : n=b
  ============================
   (nth n (Cons (tuple n b (S x)) l) (0))=(S x)

subgoal 2 is:  ...
```

La primera alternativa resulta sencilla de demostrar, sustituyendo `b` por `n`, realizando las β-reducciones y aplicando la macro `DecLeLt` definida con anterioridad.

```
RewritecL Heq2. Simpl. DecLeLt n n.
```

```
4 subgoals
  ⋮
  indH : (n,i:nat)i=n->(nth n (add i l) (0))=(S (nth n l (0)))
```

```
  n : nat
  j : nat
  Hle1 : (le a n)
  le1 : (le_lt_dec a n)=(left (le a n) (lt n a) Hle1)
  Hle2 : (le n b)
  le2 : (le_lt_dec n b)=(left (le n b) (lt b n) Hle2)
  Heq2 : ~n=b
  ============================
   (nth n (Cons (tuple n n (S x)) (Cons (tuple (S n) b x) l)) (0))=(S x)

subgoal 2 is:  ...
```

De modo similar resolvemos el resto de los casos, salvo uno en el que aplicamos la hipótesis de inducción.

```
Simpl. DecLeLt n n.
Case (eq_nat_dec n b); Intro Heq2.
RewritecL Heq2. Simpl. Rewrite le1.
DecLeLt n '(pred n). DecLeLt n n.
Simpl. Rewrite le1.
DecLeLt n n. DecLeLt n '(pred n).
Simpl. Rewrite le1. Rewrite le2.
Apply indH; Auto.
Simpl. DecLeLt n n.
Qed.
```

Con esto hemos demostrado ya la primera de las dos leyes de la especificación del algoritmo `add`. La segunda ley se plantea del siguiente modo:

```
Theorem nth_add_2: (l:seq; n, i:nat) ~i=n -> (nth n (add i l) 0) = (nth n l 0).
```

La demostración, aunque un poco más larga, es similar a la del teorema anterior, usando inducción en la secuencia de intervalos `l` y análisis por casos.

7. Forma canónica

Las secuencias de intervalos estarán representadas en todo momento siguiendo una "forma canónica", de modo que dos representaciones diferentes siempre se corresponderán con dos objetos distintos. El uso de una forma canónica nos va a proporcionar un mejor comportamiento espacial y temporal. De cara a especificar esta forma canónica introducimos una serie de definiciones inductivas, y a continuación demostraremos que la aplicación de la función `add` sobre una secuencia de intervalos de bloques en forma canónica nos proporciona un resultado que también está en forma canónica.

El predicado `ascend` nos indica si una secuencia $[(a_1, b_1, x_1), \ldots, (a_n, b_n, x_n)]$ está en orden ascendente, es decir, nos indica si $\forall i, a_i \leq b_i \wedge b_i < a_{i+1}$ es cierto o no.

```
Inductive ascend: seq -> Prop:=
 | Ascend1: (ascend Nil)
 | Ascend2: (a, b, x: nat) (le a b) -> (ascend (Cons (tuple a b x) Nil))
 | Ascend3: (a, b, x, c, d, y: nat; l:seq)
     (ascend (Cons (tuple c d y) l)) -> (le a b) -> (lt b c)
      -> (ascend (Cons (tuple a b x) (Cons (tuple c d y) l))).
```

Sin embargo, este predicado no nos llega para definir la forma canónica ya que una misma secuencia de intervalos podría representarse de varias formas. Por ejemplo, la secuencia `[(2,2,3),(3,6,3),(8,9,2)]` también se podría representar por `[(2,6,3),(8,9,2)]`, así mismo en orden ascendente y de forma más compacta. Tenemos que exigir que la representación, además de ascendente, esté "compacta", es decir, tiene que cumplirse que $\forall i, a_i \leq b_i \wedge ((x_i \neq x_{i+1} \wedge b_i < a_{i+1}) \vee (x_i = x_{i+1} \wedge b_i + 1 < a_{i+1}))$. La definición inductiva `packed` refleja el cumplimiento de las condiciones anteriores.

```
Inductive packed: seq -> Prop:=
 | Packed1: (packed Nil)
 | Packed2: (a, b, x: nat) (le a b) -> (packed (Cons (tuple a b x) Nil))
 | Packed3: (a, b, x, c, d, y: nat; l: seq)
     (packed (Cons (tuple c d y) l)) -> (le a b) -> ~x=y -> (lt b c)
      -> (packed (Cons (tuple a b x) (Cons (tuple c d y) l)))
```

```
| Packed4: (a, b, x, c, d, y: nat; l: seq)
    (packed (Cons (tuple c d y) l)) -> (le a b) -> (lt (S b) c)
     -> (packed (Cons (tuple a b x) (Cons (tuple c d y) l))).
```

El resultado devuelto por `add` va a cumplir el predicado `ascend`, pero no el predicado `packed` y por ello necesitamos definir una nueva función, `pack`, que a partir de una secuencia de intervalos de bloques con la propiedad ascendente, construya otra equivalente que además esté compactada. En la definición de `pack` usamos la función auxiliar `packAux`.

```
Fixpoint packAux [i: interval; l: seq]: seq:=
 Cases i of
 | (tuple a b x) =>
   Cases l of
   | Nil => (Cons (tuple a b x) Nil)
   | (Cons (tuple c d y) l') =>
     Cases (eq_nat_dec x y) of
     | (left _) => Cases (eq_nat_dec (S b) c) of
                   | (left _) => (packAux (tuple a d x) l')
                   | (right _) => (Cons (tuple a b x) (packAux (tuple c d y) l'))
                   end
     | (right _) => (Cons (tuple a b x) (packAux (tuple c d y) l'))
     end
   end
 end.

Definition pack [l: seq]: seq:= Cases l of Nil => Nil
                                        | (Cons i l') => (packAux i l')
                                end.
```

Para completar la forma canónica necesitamos un predicado que nos asegure que no hay en la secuencia de intervalos ningún bloque de memoria sin peticiones. En realidad, para la verificación de `add` no sería necesario chequear que para todo intervalo $(a_i, b_i, x_i), x_i \neq 0$ porque la función añade peticiones pero no las quita, pero de todos modos incluiremos la restricción `strictPosit`.

Ahora definimos `canonical` a partir de las dos últimas definiciones inductivas. Es decir, una secuencia de intervalos estará en forma canónica si está compactada y no contiene bloques de memoria sin peticiones.

```
Inductive canonical: seq -> Prop:=
  Canonical: (l: seq) (packed l) -> (strictPosit l) -> (canonical l).
```

Demostramos que la función `pack` devuelve una secuencia de intervalos equivalente a la que recibe como argumento siempre que esté en orden ascendente.

```
Lemma nth_pack: (l: seq; n: nat) (ascend l) -> (nth n l 0) = (nth n (pack l) 0).
```

La siguiente ley establece que el resultado de `add` es una secuencia de intervalos en orden ascendente.

```
Lemma ascend_add: (l:seq) (i:nat) (ascend l) -> (ascend (add i l)).
```

A continuación demostramos que el resultado de la aplicación de `pack` sobre una secuencia ascendente es una secuencia compactada.

```
Lemma packed_pack: (l: seq) (ascend l) -> (packed (pack l)).
```

También probamos que con `add` no dejamos bloques con cero peticiones.

```
Lemma strictPosit_add: (l:seq; i:nat) (strictPosit l) -> (strictPosit (add i l)).
```

Y lo mismo con la función `pack`.

```
Lemma strictPosit_pack: (l:seq) (strictPosit l) -> (strictPosit (pack l)).
```

Con esto hemos demostrado que la función `pack` nos devuelve una secuencia de intervalos "equivalente" a la que recibe como argumento y que además está en forma canónica. Por último, la demostración de que dada una secuencia en forma canónica, añadiendo una petición sobre un bloque cualquiera con `add` y aplicando a continuación `pack` obtenemos una nueva secuencia en forma canónica.

```
Theorem canonical_pack_add: (l:seq; n:nat) (canonical l) -> (canonical (pack (add n l))).
Intros l n Hpack.
Inversion_clear Hpack.
Constructor.

2 subgoals

  l : seq
  n : nat
  H : (packed l)
  H0 : (strictPosit l)
  ============================
   (packed (pack (add n l)))

subgoal 2 is:
 (strictPosit (pack (add n l)))

Apply packed_pack. Apply ascend_add.
Apply packed_ascend. Auto.
Apply strictPosit_pack.
Apply strictPosit_add. Auto.
Qed.
```

Módulo	*Líneas de código*	*Definiciones*	*Leyes*	*Proporción*	*Tamaño*
SeqDefs	49	4	0	12.25	5K
SeqLaws	144	9	5	10.29	61K
SeqCan	601	8	21	20.72	264K
totales	794	21	26	16.89	330K

Cuadro 1: Información cuantitativa del desarrollo en Coq de la verificación presentada

El cuadro 1 presenta información cuantitativa sobre cada módulo de código Coq. Las columnas corresponden respectivamente al número de líneas de código de cada teoría, a la cantidad de definiciones (incluyendo tácticas), al número de leyes, a la proporción entre el número de líneas y la cantidad de objetos definidos o probados (usado como una medida de complejidad), y el tamaño de cada módulo Coq compilado. La tabla no incluye algunos resultados de la aritmética de números naturales de la biblioteca estándar de Coq utilizados en el desarrollo.

8. Conclusiones

El sistema distribuido de vídeo bajo demanda usado para nuestro estudio representa una aplicación de un lenguaje funcional en el mundo real. Hasta aquí hemos expuesto la verificación formal de parte de la funcionalidad del servidor de forma separada. Necesitaríamos otros resultados parciales para llegar a conclusiones sobre el sistema al completo. No realizamos la verificación sobre el sistema real sino sobre un modelo del mismo.

En Coq podemos utilizar un procedimiento de extracción de programas para retirar las partes lógicas, manteniendo sólo las partes de información computacional, aunque por el momento los lenguajes proporcionados son Objetive Caml y Haskell. En este sentido, una línea de trabajo futura estaría encaminada a la creación de un proceso de "traducción" de código, limitada tal vez a algunas expresiones muy conocidas y ampliamente usadas.

La formalización de la representación canónica de las secuencias de intervalos de bloques resulta relativamente sencilla pero los problemas aparecen al definir operaciones que se complican en exceso, produciendo a continuación pruebas más complejas y tediosas. Pero desde el punto de vista algorítmico obtenemos algoritmos más eficientes.

Por último, la definición de nuevas tácticas "ad hoc", resultantes de la abstracción de patrones de prueba, nos permite eliminar algunos pasos repetitivos en el desarrollo de las pruebas cuando algunas partes se resuelven de forma similar.

Referencias

[1] J. Armstrong, R. Virding, C. Wikström, and M. Williams. *Concurrent Programming in Erlang*. Prentice-Hall, 1996.

[2] T. Arts and J. J. Sánchez. Global scheduler properties derived from local restrictions. In *ACM Sigplan Erlang Workshop at PLI'02*. ACM Press, 2002.

[3] Y. Bertot and P. Casteran. *Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions*. Springer-Verlag, 2004.

[4] R. Bird and P. Wadler. *Introduction to Functional Programming*. Prentice-Hall, 1988.

[5] T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 1988.

[6] G. Cousineau and M. Mauny. *The Functional Approach to Programming*. Cambridge University Press, 1998.

[7] G. Dowek, A. Felty, H. Herbelin, and G. Huet. The COQ proof assistant user's guide. Technical report, INRIA, 1993.

[8] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice-Hall, 1991.

[9] M. J. Gordon, A. J. Milner, and C. P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*. Springer-Verlag, 1979.

[10] V. M. Gulías, M. Barreiro, and J. L. Freire. VODKA: Developing a video-on-demand server using distributed functional programming. *Journal of Functional Programming*, 15(4), 2005.

[11] V. M. Gulías, A. Valderruten, and C. Abalde. Functional patterns for implementing distributed applications. In *IFIP/ACM Latin America Networking Conference (LANC'03)*, 2003.

[12] T. Huckle. Collection of software bugs. `http://www5.in.tum.de/~huckle/bugse.html`.

[13] P. Hudak. Conception, evolution, and application of functional programming languages. *ACM Computing Surveys*, 1989.

[14] J. S. Jorge. *Estudio de la verificación de propiedades de programas funcionales: de las pruebas manuales al uso de asistentes de pruebas*. PhD thesis, University of A Coruña, Spain, October 2004.

[15] C. Paulin-Mohring. Inductive definitions in the system Coq: Rules and properties. In *Typed Lambda Calculi and Applications*, volume 664 of *LNCS*. Springer-Verlag, 1993.

[16] L. C. Paulson. *ML for the Working Programmer*. Cambridge University Press, 2nd edition, 1996.

[17] D. A. Peled. *Software Reliability Methods*. Springer-Verlag, 2001.

[18] The Risks digest. `http://catless.ncl.ac.uk/Risks`.

[19] P. Wadler. Functional programming: An angry half dozen. *ACM SIGPLAN Notices*, 1998.

ρ_{arq} : Cálculo para el Modelamiento Formal de Arquitecturas de Software Basadas en Componentes y con Restricciones en Tiempo de Ejecución

Henry Alberto Diosa
Universidad Distrital "Francisco José de Caldas", Proyecto Curricular de Ingeniería de Sistemas,*
Bogotá D.C., Colombia
hdiosa@eisc.univalle.edu.co
Carlos Mauricio Gaona Cuevas
Universidad del Valle, Escuela de Ingeniería de Sistemas y Computación,
Cali, Colombia
mgaona@univalle.edu.co
Juan Francisco Díaz Frías
Universidad del Valle, Escuela de Ingeniería de Sistemas y Computación,
Cali, Colombia
jdiaz@univalle.edu.co

Resumen

Este artículo propone el $cálculo - \rho_{arq}$ como una aplicación del $cálculo - \rho$ [1] a la especificación de arquitecturas de software. La idea subyacente en este enfoque es partir de una configuración estática inicial que sólo es susceptible de pasar a una nueva configuración por medio de la aplicación de las reglas de la semántica operacional del $cálculo - \rho_{arq}$.

Complementariamente se propone una notación gráfica que da más alcance semántico a los diagramas de componentes en UML2.0 [18]; lo anterior, con el objeto de apoyar el diseño de configuraciones de componentes interconectados con el $cálculo - \rho_{arq}$. Esta notación permite representar aspectos dinámicos de manera consistente con propuestas de modelamiento de sistemas distribuidos como las redes de Petri [13] y el lenguaje de configuración Darwin [12].

***Palabras claves**: Arquitectura de Software, $cálculo - \rho_{arq}$, Semántica Operacional, UML2.0, Componentes, Sistemas Distribuidos*

Abstract

This paper proposes the $\rho_{arq} - calculus$ like an application of $\rho - calculus$ [1] to the specification of software architectures. The underlying idea of this focus is to leave of an initial static configuration that is only susceptible of passing to a new configuration by means of the application of the operational semantics rules of the $\rho_{arq} - calculus$.

Complementarily this approach intends a graphic notation that gives more semantic reach to the diagrams of components in UML2.0 [18]; with the above-mentioned in order to supporting the design of configurations of components interconnected with the $\rho_{arq} - calculus$. This notation allows to represent dynamic aspects in a consistent way with proposals of modeling of distributed systems as the Petri nets and the configuration language Darwin [12].

***Keywords**: Software Architecture, $\rho_{arq} - calculus$, Operational Semantics,UML2.0, Components, Distributed Systems*

1. Introducción

El modelamiento formal de arquitecturas de software es un tema de investigación abierto; este trabajo tiene la intención de mejorar notaciones informales y lenguajes descriptivos de estructuras y configuraciones arquitecturales como xADL2.0 [20] y UML2.0 [18] con expresiones formales para especificar modelos de referencia de arquitecturas orientadas a servicios, con el objeto de potenciar el chequeo de los modelos.

La similaridad sintáctica y de la semántica operacional entre el $cálculo - \rho_{arq}$ y el $cálculo - \pi$ posibilita utilizarlo para demostrar equivalencia comportamental entre sistemas constituídos por componentes de software [1] usando la observación de equivalencia propuesta por Milner para estos fines [16]. Al contar con una semántica operacional susceptible de ser traducida a Sistemas de Transición Rotulados (*STR*s, en adelante), se pueden

*Actualmente es Candidato a Ph.D. en el Doctorado en Ingenierá de la Universidad del Valle, Énfasis Ciencias de la Computación

[1] Se asimila componente de software a un autómata en el sentido de la teoría de autómatas

detección de abrazos mortales [4].

Este artículo menciona en primera instancia los trabajos relacionados con este tema (Sección 2). A continuación se presenta el $cálculo - \rho_{arq}$: sus constructos sintácticos (Sección 3) con la semántica de los mismos y la semántica operacional interpretada desde la perspectiva arquitectural (Sección 4). Posteriormente, en la Sección 5 se presenta el aporte principal de este trabajo: la especificación formal en el $cálculo - \rho_{arq}$ de aspectos arquitecturales especificados inicialmente en una notación gráfica menos formal. En la Sección 6 se presentan las conclusiones y se describen los trabajos futuros.

2. Trabajos relacionados

Trabajos recientes en métodos formales para la especificación y verificación de arquitecturas de software son los de Kaveh y Emmerich [9] que proponen verificación de correctitud y análisis de propiedades usando artefactos UML estereotipados como máquinas de estado, traducidas a un álgebra de procesos denominada FSP(*Finite State Processes*). Inverardi y Tivoli [6] proponen dos formas bisimilares de componer un sistema y luego analizan abrazos mortales y recuperación ante fallas junto con políticas de coordinación entre componentes(orquestación) usando LTL(*Lógica Lineal Temporal*). Bertolino, Inverardi y Muccini [2] proponen un enfoque para derivar pruebas de conformidad en arquitecturas de software usando STRs y FSPs. Kramer, Magee y Uchitel [7] [12] presentan un enfoque para el diseño y análisis de sistemas complejos a nivel arquitectural usando la herramienta LTSA(Labelled Transition Systems Analyzer) y Garlan [10] presenta la formalización de un lenguaje para intercambio entre descripciones arquitecturales denominado ACME.

Otro aporte interesante es el de Lamsweerde que presenta un método para mapear requerimientos a especificaciones arquitecturales de software. [3]

Sobre métodos formales aplicados al análisis de dependencia en arquitecturas de software ([8] , [5]) y sobre evaluación de desempeño a nivel arquitectural [4].

3. Sintaxis del $cálculo - \rho_{arq}$

Las entidades sintácticas se presentan en la Tabla 1, son en su mayor parte las mismas del $cálculo - \rho$ original excepto las entidades reacción interna, aplicación y se hace evidente la ausencia de celdas al asumir una perspectiva puramente declarativa y observacional de los componentes.

3.1. Símbolos

Se asume un alfabeto infinito de variables y un alfabeto infinito de nombres. Las variables son lugares para cargar nombres, es decir, no hay otros valores diferentes a los nombres. Tanto nombres como variables son indiferenciadamente denominados referencias. Términos como $\overline{x}$ equivalen a una secuencia como $(x_1, x_2, ..., x_n)$.

3.2. Expresiones

En el $cálculo - \rho_{arq}$ las expresiones representan Componentes [5] y se simbolizan por $E, F, G, \ldots$.

- La expresión $\top$ se equipara a un componente nulo que no ejecuta acción alguna.
- La composición expresa ejecución concurrente de E y F. Esta expresión será un arma poderosa para modelar componentes que se ejecutan concurrentemente en una arquitectura.
- El combinador de selección comprometida o condicionada [6] es una generalización útil del condicional, tiene la forma:
  $if\ (C_1)\ \cdots\ (C_n)\ else\ G$
  donde $C_k ::= \exists\overline{x}(\phi_k\ then\ E_k)$ con $k = 1 \ldots n$ son argumentos. Las cláusulas $(C_1)\ \cdots\ (C_n)$ pueden pensarse como computaciones en competencia, si el guarda de una cláusula es satisfecho se libera el cuerpo de la misma; si ocurre lo contrario, ésta es descartada. Si ninguna cláusula puede ser satisfecha se activa el argumento G del *else*, esto último es un camino para el manejo de fallas.

[2] En el análisis de fiabilidad(***safety property***) se busca que comportamientos indeseados no se presenten en tiempo de ejecución de un sistema

[3] En el análisis de vivacidad(***liveness property***) se deben determinar las acciones deseables que se deben ejecutar eventualmente para mantener la continuidad en la ejecución de un sistema

[4] Un abrazo mortal(***deadlock***) se define como la caída, de un sistema, a un estado del que no puede salir

[5] Para el problema de especificación que se ha abordado, un **componente** se puede considerar como un **proceso encapsulado** que sólo expone al exterior sus **puertos de entrada** y sus **puertos de salida**; siempre se podrán alambrar componentes conectando puertos de entrada de uno con los puertos de salida de otro y cuidando de que exista compatibilidad entre las **interfaces** que representan dichos puertos

[6] Esta representación de ejecución alternativa de componentes en el $cálculo - \rho_{arq}$ es una derivación de lo que se denominó **Combinador de Disyunción Condicionada o Vigilada** propuesta en las primeras versiones extendidas del $cálculo - \gamma$ [23] [24].

x, y, z	variables	
a, b, c	nombres	
$u, v, w ::= x \mid a$	referencias	
	EXPRESIONES	
$E, F, G ::=$	$\top$	**Componente Nulo**
$\mid$	$E \wedge F$	**Composición**
$\mid$	$if(C_1 \cdots C_n)\ else\ G$	**Combinador de selección comprometida**
$\mid$	$x :: \overline{y}/E$	**Abstracción**
$\mid$	$x\overline{y}/E$	**Aplicación**
$\mid$	τ/E	**Reacción interna**
$\mid$	$\exists wE$	**Declaración**
$\mid$	$x : \overline{y}/E$	**Replicación**
$\phi, \psi ::=$	$\dot{\top}$	**Verdad Lógica**
$\mid$	$\bot$	**Falsedad Lógica**
$\mid$	$x = y$	**Restricción ecuacional**
$\mid$	$\phi \dot{\wedge} \psi$	**Conjunción de restricciones**
$\mid$	$\dot{\exists}\phi$	**Cuantificador existencial**

Fuente: [23]; [1]; [24] ; [16]

Tabla 1: Sintaxis del $cálculo - \rho_{arq}$

Cuando existe una sóla cláusula no cuantificada, se tiene el condicional convencional $if\ \phi\ then\ E\ else\ G\ fi$; por esta razón, no es necesario contarlo como constructo sintáctico aparte.[7]

- La abstracción por una sola vez representa recibir una entidad simbólica a lo largo de x que podrá reemplazar a $\overline{y}$ en el componente E, siempre y cuando esa entidad simbólica recibida sea libre en el ámbito del componente E.
- La aplicación $x\overline{y}/E$ se interpreta como enviar $\overline{y}$ a lo largo de x y continuar con la ejecución de E.
- Una reacción interna representada por τ/E no tiene su contraparte explícita en el $cálculo - \rho$ original; aún así, desde la perspectiva de chequeo de modelos es importante contar con este constructo sintáctico para posibles reducciones en el número de estados a abordar en los análisis de propiedades dinámicas, donde puede requerirse especificar ciertas transiciones como reacciones internas para reducir el número de reacciones observables [2].
- La declaración $\exists wE$ introduce una referencia w con alcance E.
- La replicación $x : \overline{y}/E$ se puede expresar de la forma: $x : \overline{y}/E \equiv x :: \overline{y}/E \ \wedge\ x : \overline{y}/E$ que permite instanciar componentes; es decir, se genera una nueva abstracción lista para reaccionar y se queda listo para replicar otra si es necesario.
- Restricciones como ϕ, ψ pueden corrresponder al valor de verdadero ($\dot{\top}$), al valor de falso ($\bot$).
- Restricciones como ϕ, ψ pueden corrresponder a restricciones ecuacionales ($x = y$) con variables lógicas (es decir, sin explicitar como son obtenidos sus valores). La información sobre los valores de las variables puede establecerse a través de ecuaciones que pueden verse como restricciones. Dichas ecuaciones pueden expresar información total (por ejemplo: $x = a$) o información parcial(por ejemplo: $x = y$); recordando que los nombres son los únicos valores para variables con los que se cuenta [24].
- Restricciones como ϕ, ψ pueden corrresponder a conjunción de restricciones ($\phi \dot{\wedge} \psi$); la conjunción de restricciones es congruente a la composición de restricciones. Esto conduce a plantear que las reducciones deben ser combinadas explícitamente por reducción [1]:
- La cuantificación existencial sobre restricciones es congruente a la declaración (restricción) de variables sobre restricciones ($\exists x\phi$)

[7]Este constructo introduce no-determinismo cuando se cuenta con más de una cláusula que puede ser satisfecha desde el contexto. En el caso del $cálculo - \rho_{arq}$ se pueden establecer valores a las variables a través del contexto para controlar reacciones entre componentes interactuantes y se puede modelar selección alternativa de componentes para lograr comunicación por medio de la activación(selección) de enlaces usando el combinador de selección comprometida; el cual puede permitir el manejo de fallas, ilustrado de manera precisa en [23]pp.20-21. Esta forma de controlar el no-determinismo en el $cálculo - \rho_{arq}$ puede ser una herramienta valiosa para la reducción del número de posibles estados en una etapa de análisis de propiedades comportamentales y puede fortalecer la propiedad de fiabilidad en un arquitectura

El $cálculo - \rho_{arq}$ es sintácticamente composicional porque restricciones, aplicaciones y condicionales pueden combinarse libremente por medio de composición, declaración y abstracción. La relación de reducción de este cálculo se define de la misma manera que se hace en el $cálculo - \rho$ [1]y en el $cálculo - \gamma$ [24].

Ahora, dada la presencia de restricciones, se deben tener en cuenta las definiciones propuestas por Niehren y Muller en [1]. La Tabla 2 presenta los axiomas de congruencia estructural; variables ligadas son introducidas como argumentos formales de las abstracciones y por las declaraciones. [8]

$(\alpha - conversión)$	Cambio de referencias ligadas por referencias libres
(ACI)	$\wedge$ es asociativa, conmutativa y satisface $E \wedge \top \equiv E$
$(Intercambio)$	$\exists x \exists y E \equiv \exists y \exists x E$
$(Alcance)$	$\exists x\ E \wedge F \equiv \exists x(E \wedge F) \quad si\ x \notin \mathcal{FV}(F)$
$(Equiv.Restricciones)$	$\phi \equiv \psi \quad si\ \phi \models\!\!\!\models_\Delta \psi$ y $\mathcal{FV}(\phi) = \mathcal{FV}(\psi)$

Fuente [1]

Tabla 2: Congruencia Estructural $Cálculo - \rho_{arq}$

La Tabla 3 presenta las reglas de reducción que representan la semántica operacional. El operador de reemplazo $[\overline{z}/\overline{y}]$ requiere implícitamente que $\overline{z}$ y $\overline{y}$ tengan la misma longitud y que $\overline{y}$ sea lineal, es decir, que todos los elementos de $\overline{y}$ sean distintos entre sí.

$(A_{\rho_{arq}})\quad \phi \wedge x : \overline{y}/E \wedge x'\overline{z}/F \longrightarrow \phi \wedge x : \overline{y}/E \wedge [\overline{z}/\overline{y}]E \wedge F$	$si\ \phi \models_\Delta x = x', \mathcal{V}(\overline{z}) \cap \mathcal{BV}(E) = \emptyset$
$(C_{\rho_{arq}})\quad \phi_1 \wedge \phi_2 \longrightarrow \psi$	$si\ \phi_1 \dot{\wedge} \phi_2 \models\!\!\!\models_\Delta \psi$
$(Then_{\rho_{arq}})\quad \phi \wedge if\ \psi\ then\ E\ else\ F\ fi \longrightarrow \phi \wedge E$	$si\ \phi \models_\Delta \psi$
$(Else_{\rho_{arq}})\quad \phi \wedge if\ \psi\ then\ E\ else\ F\ fi \longrightarrow \phi \wedge F$	$si\ \phi \models_\Delta \neg\psi$

Fuente [1]

Tabla 3: Reglas de Reducción del $Cálculo - \rho_{arq}$

Estas tres reglas de reducción, desde la perspectiva arquitectural, se pueden asociar a restricciones del contexto global de la arquitectura de referencia y controlan (si son suficientemente fuertes) el comportamiento de componentes/conectores de la misma en tiempo de ejecución.

La regla de reducción de combinación de restricciones (C_ρ) merece especial atención porque restricciones a nivel global de la arquitectura podrían obtener más información o ampliarse por este medio (Operador $Tell$, tal como se propone en el modelo de computación por restricciones [22]).

4.1. Reducciones en componentes

Ahora, luego de acceder a los agentes $E, F, G, ...$(satisfacer las restricciones) se podrían dar reducciones que respetan las reglas de inferencia siguientes:

$$\frac{E \longrightarrow F}{\exists x E \longrightarrow \exists x F} \qquad \frac{E \longrightarrow F}{E \wedge G \longrightarrow F \wedge G} \qquad \frac{E_1 \equiv E_2 \quad E_2 \longrightarrow F_2 \quad F_2 \equiv F_1}{E_1 \longrightarrow F_1}$$

Tabla 4: Reducciones sobre los agentes o componentes

4.2. Sobre la ausencia de estado en el $Cálculo - \rho_{arq}$

Ni en los constructos sintácticos ni en la semántica operacional del $cálculo - \rho_{arq}$ se usa el concepto de celda introducido en el $cálculo - \rho$ original. Esto es consecuencia de considerar un componente como una entidad sin estado, que se ciñe más a un modelo de computación declarativo que hace una vista observacional, regida por principios de abstracción que modelan un componente como una caja negra que recibe entradas y garantiza producir las mismas salidas ante los mismos valores de dichas entradas.

[8] Las variables que no están ligadas se denomina variables libres. $\mathcal{FV}(E)$ y $\mathcal{BV}(E)$ se usan para denotar el conjunto de variables libres y variables ligadas, respectivamente, en E

Se propone a continuación una notación gráfica consistente y traducible al $cálculo - \rho_{arq}$ que se inspira en tres fuentes principales:

1. La notación gráfica de componente de software propuesta en UML2.0 [18], haciéndola más formal respecto al tema de arquitecturas de software y traducible a la especificación formal en el $cálculo - \rho_{arq}$.
2. El enfoque de modelamiento de sistemas distribuídos usando redes de Petri ordinarias para potenciar las posibilidades de análisis de propiedades dinámicas [13].
3. La propuesta de traducción de la notación visual del lenguaje de configuración Darwin al $cálculo - \pi$ [12].

5.1. La Notación de Componente

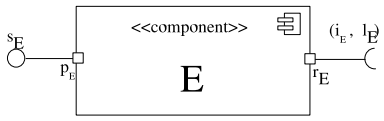


Figura 1: Notación Gráfica de Componente

Un componente se representa como lo muestra la Figura 1. Cuadrados que sobresalen de los bordes representan puertos públicos [9] y poseen un nombre de acceso con el subíndice indicando el componente al que pertenece; los puertos se pueden interpretar como los puntos o nombres de acceso que se usan para acceder a un servicio ofrecido o a una locación que requiere alguno, en el ejemplo de la Figura 1 se usan las letras p y r con un subíndice que indica el componente correspondiente (ejemplo: p_E). A cada servicio sólo corresponderá un nombre de acceso y si hay más de un servicio ofrecido por el componente se puede igualmente numerar el subíndice(ejemplo: p_{E1}); un componente puede tener varios puertos con una interfaz de entrada o de salida asociada a cada uno.

5.1.1. Interfaz de salida (Provided Interface)

Una interfaz de salida se representa con una línea recta continua saliendo de un puerto y terminando en un círculo cerrado no relleno; dicha interfaz de salida se denominará también lugar de salida o de provisión de servicio. Éste tendrá el nombre o referencia del servicio que se provee identificado por una s con un subíndice que indica el componente al que corresponde (ejemplo: s_E), si hay más de un servicio a referenciar se puede numerar en el subíndice (ejemplo: s_{E1}).

Especificación formal en el $cálculo - \rho_{arq}$**.** Para el componente E que se presenta en la Figura 1 se denomina $PROV_E(p, s)$ y se define formalmente por:

$PROV_E(p, s) \stackrel{def}{=} p_E : x/xs_E \equiv p_E :: x/xs_E \ \wedge \ p_E : x/xs_E$

se interpreta como: *se espera a lo largo de* p_E *un valor de una locación a la cual se enviará el servicio* s_E; al usar replicación es posible atender muchos clientes que requieran este servicio.

5.1.2. Interfaz de entrada (Required Interface)

Se representa por una línea recta continua saliendo del rectángulo y terminada en un semicírculo abierto; también se denominará lugar requisitor de servicio o lugar de entrada, éste se rotulará con una pareja como (i_E, l_E) que indica una ubicación l_E que espera recibir un valor que pueda reemplazar el parámetro i_E en el componente E.

Especificación formal en el $cálculo - \rho_{arq}$**.** Para el componente que se presenta en la Figura 1 se denomina $REQ_E(r, l)$ y se define por:

$REQ_E(r, l) \stackrel{def}{=} r_E :: y/yl_E$

que se puede interpretar como: *se espera a lo largo de* r_E *un nombre de acceso a un servicio, que al ser aplicado a la ubicación* l*, conecte ésta con el servicio.*

Por consiguiente, desde la perspectiva de definición de componente, se puede plantear que sería suficientemente representado por sus interfaces públicas de entrada y salida actuando concurrentemente con la abstracción que represente la incorporación de servicios al mismo, es decir, hasta ese punto habría visibilidad para un observador del componente. En el caso que se viene ejemplificando, el componente E se puede definir como:

$E \stackrel{def}{=} PROV_E(p, s) \ \wedge \ REQ_E(r, l) \ \wedge \ l_E :: i_E/E$

[9] Si un puerto es protegido, se colocará el cuadrado en la parte interna de la frontera del rectángulo

Un componente adquiere utilidad práctica cuando interactúa con otros componentes, para esto debe ocurrir que servicios ofrecidos en un lugar sean trasladados a la ubicación que los requiere. La notación gráfica que representará, de ahora en adelante, el ensamble de componentes será similar a la que ilustra la Figura 2. En esta configuración el conector de ensamble [18] enlaza activamente la locación l_E y el servicio s_F. Esta situación se representa con la presencia de un círculo relleno dentro del círculo del lugar de provisión del servicio, dicho círculo relleno se asemeja a un ***token*** que indica que se ha entregado el servicio en esa ubicación.

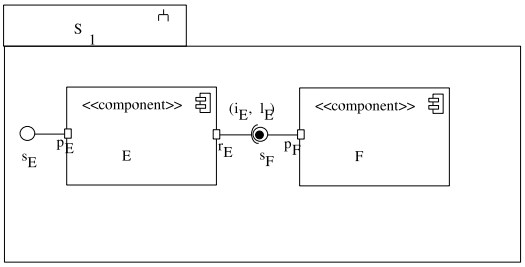


Figura 2: Notación Gráfica de Ensamble de Componentes

Como se puede observar en la Figura 2, se tienen dos componentes:

$E \stackrel{def}{=} (p_E : x/xs_E) \wedge (r_E :: y/yl_E) \wedge (l_E :: i_E/E)$
$F \stackrel{def}{=} (p_F : z/zs_F)$

Especificación formal en el $cálculo - \rho_{arq}$**.** Para conectar activamente E y F se requiere un conector que logre dar acceso al servicio; este conector de ensamble se denominará C_{FE} y se especifica como:
$C_{FE} \stackrel{def}{=} r_E p_F$
que al actuar en forma concurrente(composición), hará que el sistema S_1 representado por esta configuración inicie una dinámica de ejecución, que se puede especificar así:
$S_1 = E \wedge F \wedge C_{FE}$
$= \{(p_E : x/xs_E) \wedge (r_E :: y/yl_E) \wedge (l_E :: i_E/E)\} \wedge \{(p_F : z/zs_F)\} \wedge \{r_E p_F\}$
Ahora si se aplican los axiomas de congruencia estructural y las reglas de reducción del $cálculo - \rho_{arq}$ sin presencia de restricciones (que se incorporarán más adelante), se tiene[10]:

$S_1 \stackrel{A_{\rho arq}}{\Longrightarrow} p_E : x/xs_E \wedge [s_F/i_E]E$

Lo anterior muestra el proceso de ejecución que lleva el servicio(*token*) s_F a la ubicación l_E que lo requiere, es decir, el flujo del *token* indica la secuencia de ejecución en la arquitectura; surge entonces la necesidad de expresar esta secuencia de ejecución.

5.3. Control de ejecución arquitectural

El flujo de *tokens* o servicios en una arquitectura basada en componentes representa su dinámica en tiempo de ejecución; en la anterior sección, se pudo observar que al ejecutar F se provee un servicio[11] que es requerido por E; ahora, si E está conectado a su vez a un componente que requiere de sus servicios, se requeriría especificar formalmente el flujo de ejecución. Al considerar E como un componente, no podemos observar más allá de sus entradas y salidas, el flujo de ejecución será especificado a partir de este nivel de observabilidad; por consiguiente, en el ejemplo ilustrativo que nos ocupa, cuando se llega a $[s_F/i_E]E$ lo único que se podría detectar es que el componente provea el servicio normalmente(genera el *token* en sus puntos de salida) o que indique una falla en su ejecución. Si se produce el *token* en su(s) punto(s) de salida o de provisión de servicio, se especifica activando el conector correspondiente para continuar la ejecución de los componentes subsiguientes; en otro caso, se debe entregar un mensaje de error a un componente manejador de errores. Una forma de expresar esta situación es usando un condicional de la forma:

[10]Para evitar verbosidad formal y debido al limitado espacio en el documento se introduce el símbolo relación $\stackrel{A_{\rho arq}}{\Longrightarrow} \equiv \stackrel{A_{\rho arq}}{\longrightarrow} \ldots \stackrel{A_{\rho arq}}{\longrightarrow}$. Para un lector interesado en el desarrollo completo, puede solicitar a los autores el reporte técnico referenciado en [11]

[11]Se debe tener en cuenta que un servicio obtenido se ve representado normalmente por una instancia de un tipo de dato que entra al componente que lo requirió para propósitos específicos de ejecución

else (*Activar conector de manejo de errores*) fi

Se retoma el ejemplo ilustrativo que se viene trabajando, ampliándolo con un posible componente T que requiere un servicio de E para ejecutarse y un componente manejador de errores (Ver Figura 3). Cada componente tiene como especificación en el $cálculo - \rho_{arq}$:

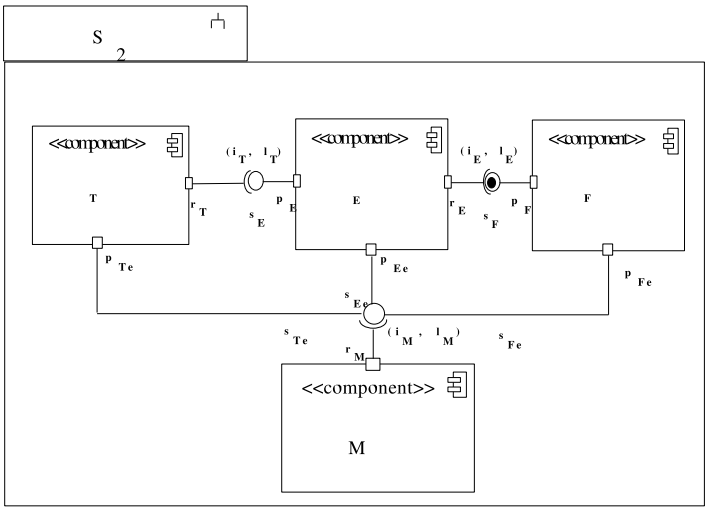


Figura 3: Ensamble Complejo de Componentes. Momento Inicial: Captura de Datos Exitosa

$F \stackrel{def}{=} (p_F : z/zs_F) \wedge (p_{Fe} : w/ws_{Fe})$
$E \stackrel{def}{=} (p_E : x/xs_E) \wedge (r_E :: y/yl_E) \wedge (p_{Ee} : v/vs_{Ee}) \wedge (l_E :: i_E/E)$
$M \stackrel{def}{=} (r_M : y/yl_M) \wedge (l_M : i_M/M)$
$T \stackrel{def}{=} (r_T :: q/ql_T) \wedge (l_T : i_T/T) \wedge (p_{Te} : n/ns_{Te})$

Antes de proseguir se introducen dos conceptos de componente que serán necesarios en lo que sigue:

- **COMPONENTE FUENTE**. Es aquel que sólo posee interfaces de salida o de provisión de servicio; en nuestro ejemplo, el componente F.
- **COMPONENTE SUMIDERO**. Es aquel que sólo posee interfaces de entrada o requisitoras de servicio. En nuestro ejemplo, es de este tipo el componente M o manejador de errores.

5.3.1. La regla τ o de éxito/fracaso observacional

Dado que el interés de este trabajo no va más allá de especificar arquitecturas de software como ensamble de componentes; cualquier entrada a un componente que se hace efectiva mediante el reemplazo de sus parámetros de entrada por valores obtenidos como servicios provistos por otros componentes, se expresa como $[v/w]E$ (v *que reemplaza a* w *en el componente* E con $v \notin \mathcal{VF}(E)$), la representación del éxito o fracaso de la ejecución se hará usando la regla de reducción denominada τ o de **Éxito/Fracaso Observacional** porque es producto de lo que ocurre internamente en el componente y que no es visible al observador externo del mismo:

$[v/w]E \longrightarrow E^{\top}$; si hay ejecución exitosa del componente
$[v/w]E \longrightarrow E^{\perp}$; si falla la ejecución del componente.

De esta manera se podría representar la ejecución exitosa de un componente por $E^{\top}$ y su ejecución fallida por $E^{\perp}$; a partir de la regla de reducción τ y la perspectiva puramente observacional, se pueden proponer dos nuevos axiomas de congruencia estructural:

$E^{\top} \equiv \dot{\top}$; si y solo si $[v/w]E \longrightarrow E^{\top}$
y
$E^{\perp} \equiv \perp$; si y solo si $[v/w]E \longrightarrow E^{\perp}$
Estas nuevas reglas de reducción y axiomas permiten especificar formalmente el avance en la ejecución de una arquitectura; para corroborar esto se vuelve al ejemplo del sistema S_2, cuya configuración inicial se presenta en

$S_2 = \{if\ F^\top\ then\ F \wedge C_{FE} \wedge E\ else\ C_{FM} \wedge M\} \wedge \{if\ E^\top\ then\ C_{ET} \wedge T\ else\ C_{EM} \wedge M\} \wedge \{if\ T^\top\ then\ S_2 = éxito\ else\ C_{TM} \wedge M\}$
Inicialmente se supone que todos los componentes trabajan correctamente, es decir, hay éxito en su ejecución, se obtendría en primera instancia que el componente F es exitoso:[12]

$$S_2 \xrightarrow{\tau} \{F \wedge C_{FE} \wedge E\} \wedge \{if\ E^\top\ then\ C_{ET} \wedge T\ else\ C_{EM} \wedge M\} \wedge \{if\ T^\top\ then\ S_2 = éxito\ else\ C_{TM} \wedge M\}$$

efectuando los reemplazos correspondientes, se obtiene:

$$S_2 = \{[(p_F : z/zs_F) \wedge (p_{Fe} : w/ws_{Fe})] \wedge [r_E p_F] \wedge [(p_E : x/xs_E) \wedge (r_E :: y/yl_E) \wedge (p_{Ee} : v/vs_{Ee}) \wedge (l_E :: i_E/E)]\} \wedge \{if\ E^\top\ then\ C_{ET} \wedge T\ else\ C_{EM} \wedge M\} \wedge \{if\ T^\top\ then\ S_2 = éxito\ else\ C_{TM} \wedge M\}$$

$$\xRightarrow{A_{parq}} \{[(p_F : z/zs_F) \wedge (p_{Fe} : w/ws_{Fe})] \wedge [(p_E : x/xs_E) \wedge (p_{Ee} : v/vs_{Ee}) \wedge ([s_F/i_E]E)]\} \wedge \{if\ E^\top\ then\ C_{ET} \wedge T\ else\ C_{EM} \wedge M\} \wedge \{if\ T^\top\ then\ S_2 = éxito\ else\ C_{TM} \wedge M\}$$

esta primera fase de ejecución coloca el servicio s_F listo para reemplazar el parámetro de entrada i_E en el componente E, lo cual genera ejecución de dicho componente por tener copadas sus entradas; luego, se echa mano de la regla de reducción τ bajo la suposición que hay ejecución exitosa:

$$S_2 \xrightarrow{\tau} \{[(p_F : z/zs_F) \wedge (p_{Fe} : w/ws_{Fe})] \wedge [(p_E : x/xs_E) \wedge (p_{Ee} : v/vs_{Ee}) \wedge (\dot{\top})]\} \wedge \{C_{ET} \wedge T\} \wedge \{if\ T^\top\ then\ S_2 = éxito\ else\ C_{TM} \wedge M\}$$

aplicar la nueva secuencia de reducciones lleva el sistema a una nueva configuración, la mostrada en la Figura 4, donde el *token* se ha desplazado al punto de entrada del componente T[13]; dicha secuencia(haciendo los reemplazos correspondientes) sería:

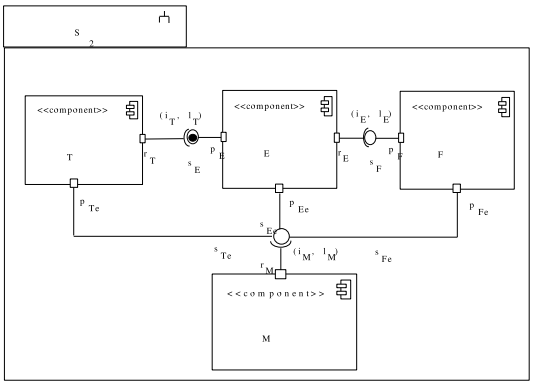


Figura 4: Ensamble Complejo de Componentes. Momento:Activación Conector Entre E y T

$$S_2 = \{[(p_F : z/zs_F) \wedge (p_{Fe} : w/ws_{Fe})] \wedge [(p_E : x/xs_E) \wedge (p_{Ee} : v/vs_{Ee})]\} \wedge \{[r_T p_E] \wedge [(r_T :: q/ql_T) \wedge (l_T : i_T/T) \wedge (p_{Te} : n/ns_{Te})]\} \wedge \{if\ T^\top\ then\ S_2 = éxito\ else\ C_{TM} \wedge M\}$$

$$\xRightarrow{A_{parq}} \{[(p_F : z/zs_F) \wedge (p_{Fe} : w/ws_{Fe})] \wedge [(p_E : x/xs_E) \wedge (p_{Ee} : v/vs_{Ee})]\} \wedge \{[(l_T : i_T/T) \wedge ([s_E/i_T]T) \wedge (p_{Te} : n/ns_{Te})]\} \wedge \{if\ T^\top\ then\ S_2 = éxito\ else\ C_{TM} \wedge M\}$$

la expresión $([s_E/i_T]T)$ indica que se ha provisto el servicio s_E que reemplaza el parámetro de entrada i_T en el componente T. Ahora, el componente T puede ser exitoso, es decir, proveer la funcionalidad requerida al usuario final o fallar, entregar un mensaje al componente manejador de errores. En el primer caso, será visible una ejecución exitosa del sistema S_2 por medio del componente T; en caso contrario, se activará el conector que enlaza el componente con el manejador de errores para proveer a éste del servicio de información de errores s_{Te}, que simplemente puede ser un mensaje de error al mismo. En el segundo escenario de error, se obtendría:

[12] Al ser F un componente fuente, se puede asimilar a una interfaz de captura de datos certificada, es decir, que garantiza un funcionamiento correcto

[13] Un lector atento podrá observar que E en la especificación formal se inhabilita para recibir por sus puntos de entrada; esto se debe a que se usó aplicación por una sola vez en vez de replicación para los puertos de entrada

$y/yl_E) \wedge (p_{Ee} : v/vs_{Ee}) \wedge (l_E :: i_E/E)] \wedge [(l_T : i_T/T) \wedge (\bot) \wedge (p_{Te} : n/ns_{Te})]\} \wedge \{ C_{TM} \wedge M\}$

$\equiv \{[(p_F : z/zs_F) \wedge (p_{Fe} : w/ws_{Fe})] \wedge [(p_E : x/xs_E) \wedge (p_{Ee} : v/vs_{Ee})] \} \wedge \{[(p_E : x/xs_E) \wedge (r_E :: y/yl_E) \wedge (p_{Ee} : v/vs_{Ee}) \wedge (l_E :: i_E/E)] \wedge [(l_T : i_T/T) \wedge (\bot) \wedge (p_{Te} : n/ns_{Te})]\} \wedge \{ r_M p_{Te} \wedge [(r_M : y/yl_M) \wedge (l_M : i_M/M)]\}$

$S_2 \stackrel{A_{\rho arq}}{\Longrightarrow} \{[(p_F : z/zs_F) \wedge (p_{Fe} : w/ws_{Fe})] \wedge [(p_E : x/xs_E) \wedge (p_{Ee} : v/vs_{Ee})] \} \wedge \{[(p_E : x/xs_E) \wedge (r_E :: y/yl_E) \wedge (p_{Ee} : v/vs_{Ee}) \wedge (l_E :: i_E/E)] \wedge [(l_T : i_T/T) \wedge (\bot) \wedge (p_{Te} : n/ns_{Te})]\} \wedge \{ [(r_M : y/yl_M) \wedge (l_M : i_M/M) \wedge ([s_{Te}/i_M]M)]\}$

si se observa con atención, la interfaz de entrada al manejador de errores se replica para seguir atendiendo nuevos envíos de errores.

La regla de reducción τ es útil para modelar la secuencia de ejecución en una arquitectura basada en componentes. Por consiguiente, si un componente tiene varias interfaces de entrada no podría ejecutarse hasta obtener todos los servicios de éstas o una combinación determinada por el arquitecto de software; en este caso, se establece como guarda de la activación del conector, o conectores de salida, *la conjunción de las aplicaciones exitosas a través de las cuales se reemplazan los parámetros de entrada.*

5.4. Configuración jerárquica o componentes compuestos

Ahora se abordarán configuraciones donde existen *mapeos* desde interfaces internas a interfaces externas porque un componente es compuesto.

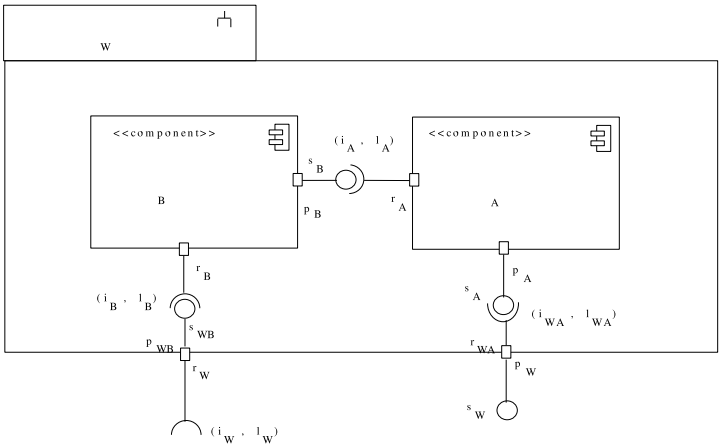


Figura 5: Mapeo de interfaces internas

5.4.1. Especificación formal en el cálculo $-\ \rho_{arq}$

Para formalizar estas configuraciones arquitecturales se deben considerar los puertos como componentes sin procesamiento interno pero con capacidad de redireccionar o reemplazar un servicio interno por uno obtenido desde lo externo, es decir, no aplica a éstos la regla τ o de éxito/fracaso observacional; sin embargo, se introduce un término de redireccionamiento y de activación por defecto del conector de ensamble entre el puerto externo y el interno; por esta razón, cuenta con cuatro parámetros representando información interna y externa al componente. Formalmente estarán definidos por una composición entre el formalismo que expresa una interfaz de requerimiento de servicio y una interfaz de provisión de servicio. Así, en la configuración ilustrada en la Figura 5, se definirían como:

$$PROV_{jer}(p_W, s_W, r_{WA}, l_{WA}) \stackrel{def}{=} r_{WA} : x/xl_{WA} \wedge l_{WA} :: s_W/(p_W : x/xs_W)$$
$$\equiv r_{WA} :: x/xl_{WA} \wedge l_{WA} :: s_W/(p_W : x/xs_W)$$

y:

$$REQ_{jer}(r_W, l_W, p_{WB}, s_{WB}) \stackrel{def}{=} r_W :: y/yl_W \wedge l_W :: s_{WB}/(p_{WB} : x/xs_{WB}) \wedge r_B p_{WB}$$
$$\equiv r_W :: y/yl_W \wedge l_W :: s_{WB}/(p_{WB} : x/xs_{WB}) \wedge r_B p_{WB}$$

El componente compuesto W se podría especificar como:

$PROV_{jer}(p_W, s_W, r_{WA}, l_{WA})$

Se puede demostrar que si W es un componente más de una configuración dada y recibe un *token* en su interfaz de requerimiento de servicio, éste hara fluir el servicio hasta su interfaz de salida.

5.5. Configuración de componentes alternativos o condicionados

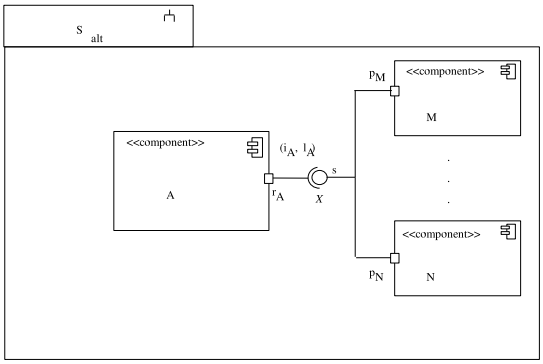


Figura 6: Configuración con Componentes Alternativos para Proveer Servicios

Una de las posibles configuraciones en arquitecturas de software orientadas a servicios es que exista un grupo de componentes que ofrecen el mismo servicio y de los cuales, un componente requisitor, puede escoger uno de acuerdo a restricciones arquitecturales establecidas desde el contexto en tiempo de ejecución. En la Figura 6 el componente sumidero A puede recibir el servicio desde dos proveedores(componentes fuentes): M o N, la variable X en el conector de ensamble indica que el valor que ésta cargue determinará cuál de las dos opciones será tomada.

5.5.1. Especificación formal en el cálculo $-\ \rho_{arq}$

Para este tipo de configuraciones será de gran utilidad el combinador de selección comprometida. La especificación formal sería:

$S_{alt} \stackrel{def}{=} [if\ \exists X((X = p_M)\ then\ \{if\ M^{\top}\ then\ r_A p_M\ \wedge\ M\ \})\ \cdots\ ((X = p_N)\ then\ \{if\ N^{\top}\ then\ r_A p_N\ \wedge\ N\ \})\ else\ Manejarerror] \wedge A$

las expresiones formales para los componentes serían:

$A \stackrel{def}{=} REQ_A(r_A, l_A) \wedge l_A :: i_A/A \equiv r_A :: y/yl_A \wedge l_A :: i_A/A$
$M \stackrel{def}{=} PROV_M(p_M, s) \equiv p_M : x/xs$
$N \stackrel{def}{=} PROV_N(p_N, s) \equiv p_N : x/xs$

Ahora, se supone un escenario donde el contexto determina que el componente a escoger es M y no hay error, es decir, $X = p_M$ y $M^{\top} \equiv \dot{\top}$:
$(X = p_M \wedge S_{alt}) \equiv \{ r_A p_M \wedge M \} \wedge A$
observe que la expresión entre paréntesis, a la izquierda, se corresponde con la regla de reducción $Then_{\rho_{arq}}$ que corresponde simplemente a un ASK a un *Global Store* de restricciones; que podría ser el medio para gestionar la arquitectura en tiempo de ejecución. Continuando con el ejemplo, se hacen los reemplazos correspondientes:

$(X = p_M \wedge S_{alt}) \equiv \{ r_A p_M \wedge p_M : x/xs \} \wedge r_A :: y/yl_A \wedge l_A :: i_A/A$
$(X = p_M \wedge S_{alt}) \stackrel{A_{\rho_{arq}}}{\longrightarrow} \{ p_M : x/xs \} \wedge p_M l_A \wedge l_A :: i_A/A$
$(X = p_M \wedge S_{alt}) \stackrel{A_{\rho_{arq}}}{\longrightarrow} \{ p_M : x/xs \} \wedge l_A s \wedge l_A :: i_A/A$
$(X = p_M \wedge S_{alt}) \stackrel{A_{\rho_{arq}}}{\longrightarrow} \{ p_M : x/xs \} \wedge [s/i_A]A$

reemplazado por el servicio s provisto por componentes alternativos. De manera análoga se podría hacer para activar el componente N en vez de M u otros componentes alternativos.

6. Conclusiones y trabajo futuro

Una de las razones de la formulación del $cálculo - \rho_{arq}$ es la existencia de una máquina abstracta para el lenguaje de programación Mozart, cuyo fundamento formal es el $cálculo - \rho$, lo que permitirá soportar aspectos de implementación de manera natural y dentro de un marco de trabajo multiparadigma para implementar componentes. [14].

Se abordará ahora la traducción de configuraciones básicas, como las descritas aquí, desde documentos xADL2.0 [20] al $cálculo - \rho_{arq}$; lo anterior, con el objetivo de contar con una herramienta de especificación estática que sea portable y abierta. Al lograr este paso se abre el camino para desarrollar una herramienta que genere la traducción de manera automática, tanto al cálculo formal como a la notación gráfica; esto permitirá brindar una herramienta útil al arquitecto de software para manipular configuraciones de componentes definidas de manera gráfica y textual, con la potencialidad de un cálculo de procesos incorporando restricciones para analizar aspectos dinámicos.

Algo que puede ser novedoso y útil en el tratamiento de especificaciones arquitecturales es que el uso de restricciones permite condicionar la participación de componentes de software en posibles instancias arquitecturales ejemplificadas desde un modelo de referencia; esto a su vez posibilitaría activar o desactivar componentes de software en tiempo de ejecución desde un contexto controlado por restricciones almacenadas en un repositorio global. Esta última característica en este enfoque de especificación puede abrir las puertas a la gestión de arquitecturas basadas en componentes, a partir de un modelo de computación por restricciones; se abre la perspectiva para que conceptos como el de Global Store [21] en Mozart, que se derivan del sustrato formal que lo soporta($cálculo - \rho$), podrán permitir la implementación real de arquitecturas con estas posibilidades.

La notación gráfica mejorada de UML 2.0 [18] se acomoda muy bien a la teoría de redes de Petri [13]; recordemos que una red de Petri es una clase particular de grafo dirigido y ponderado con dos clases de nodos denominados lugares y transiciones, con arcos que van desde un lugar a una transición o desde una transición a un lugar. En la representación gráfica usada aquí los lugares se corresponden con los conectores de ensamble que usan círculos que pueden cargar un *token* indicando flujo del servicio a través del mismo. Las *transiciones* se asocian a componentes o lugares de computación en sí mismos.

En el momento de chequeo de modelos será de gran utilidad el contar no sólo con la teoría de observabilidad de equivalencia inherente a las álgebras de procesos sino con la potencia analítica de aspectos dinámicos de modelos usando redes de Petri. Se concluye que esta es una afortunada coincidencia y está en consonancia con recientes aproximaciones teóricas como la de Milner con su propuesta de teoría de bigrafos[14][17] relacionadas últimamente con redes de Petri [15] [19].

Referencias

[1] Asian Computer Science Conference ACSC'95. *Constraints for Free in Concurrent Computation* (1995).

[2] Bernardo, M., and Inverardi, P., Eds. *Formal Methods in Testing Software Architectures* (sep 2003), Lecture Notes in Computer Science. Advanced Lectures, Third International School on Formal Methods for the Design of Computer, Communication and Software Systems:Software Architectures, Springer.

[3] Bernardo, M., and Inverardi, P., Eds. *From System Goals to Software Architecture* (sep 2003), Lecture Notes in Computer Science. Advanced Lectures, Third International School on Formal Methods for the Design of Computer, Communication and Software Systems:Software Architectures, Springer.

[4] Bernardo, M., and Inverardi, P., Eds. *Performance Evaluation at the Software Architecture Level* (sep 2003), Lecture Notes in Computer Science. Advanced Lectures, Third International School on Formal Methods for the Design of Computer, Communication and Software Systems:Software Architectures, Springer.

[5] Bernardo, M., and Inverardi, P., Eds. *Software Architecture and Dependability* (sep 2003), Lecture Notes in Computer Science. Advanced Lectures, Third International School on Formal Methods for the Design of Computer, Communication and Software Systems:Software Architectures, Springer.

[14] Existe un cálculo formal que extiende el $cálculo - \pi$ de Milner [16] con restricciones:el $cálculo - \pi^+$[25]. Éste posee dos constructos sintácticos muy interesantes: los agentes restricción $Tell$ y Ask, el primero puede reemplazar a la regla de reducción $C_{\rho_{arq}}$ y el segundo a las reglas $Then_{\rho_{arq}}$ y $Else_{\rho_{arq}}$, haciendo más sencillas y compactas las expresiones

2003), Lecture Notes in Computer Science. Advanced Lectures, Third International School on Formal Methods for the Design of Computer, Communication and Software Systems:Software Architectures, Springer.

[7] Bernardo, M., and Inverardi, P., Eds. *Software Architecture Modeling '&Ánalysis: A Rigorous Approach* (sep 2003), Lecture Notes in Computer Science. Advanced Lectures, Third International School on Formal Methods for the Design of Computer, Communication and Software Systems:Software Architectures, Springer.

[8] Bernardo, M., and Inverardi, P., Eds. *The Application of Dependence Analysis to Software Architecture Descriptions* (sep 2003), Lecture Notes in Computer Science. Advanced Lectures, Third International School on Formal Methods for the Design of Computer, Communication and Software Systems:Software Architectures, Springer.

[9] Bernardo, M., and Inverardi, P., Eds. *Validating Distributed Object and Component Designs* (sep 2003), Lecture Notes in Computer Science. Advanced Lectures, Third International School on Formal Methods for the Design of Computer, Communication and Software Systems:Software Architectures, Springer.

[10] CASCON'97. *ACME:An Architecture Description Interchange Language* (Nov. 1997).

[11] Diosa, H. A. Cálculo formal para especificar aspectos dinámicos de arquitecturas de software con componentes condicionados. Tech. rep., Escuela de Ingeniería de Sistemas y Computación. Universidad del Valle, Cali,Colombia, May 2005.

[12] Fifth European Software Engineering Conference. *Specifying Distributed Software Architectures* (Sept. 1995).

[13] IEEE. *Petri Nets: Properties, Analysis and Applications* (Apr. 1989), vol. 77.

[14] Jensen, O. H., and Milner, R. Bigraphs and mobile processes(revised). Tech. rep., University of Cambridge. Computer Laboratory, feb 2004.

[15] Leifer, J. J., and Milner, R. Transitions systems, link graphs and Petri nets. Tech. rep., University of Cambridge. Computer Laboratory, aug 2004.

[16] Milner, R. *Communicating and Mobile Systems:the π-Calculus.* Cambridge University Press, 1999.

[17] Milner, R. Bigraphs whose names have locality multiple. Tech. rep., University of Cambridge. Computer Laboratory, sep 2004.

[18] Object Management Group. UML2.0 Superstructure Specification, oct 2004.

[19] Raymond Devillers/, H. K., and Koutny, M. A Petri net translation of pi-calculus terms. Tech. rep., School of Computing Science,University of Newcastle upon Tyne, jan 2005.

[20] Richard N. Taylor A. v. d. H. y. E. M. D. An Infrastructure for the Rapid Development of XML-based Architecture Description Languages. *ACM ICSE'02 11*, 1 (May 2002), 266–276.

[21] Roy, P. V., and Haridi, S. *Concepts, Techniques and Models of Computer Programming.* MIT Press, 2004.

[22] Saraswat, V. A. *Concurrent Constraint Programming.* The MIT Press, 1992.

[23] Smolka, G. A Calculus for Higher-order Concurrent Constraint Programming with Deep Guards. Tech. rep., Bundesminister fr Forschung und Technologie, 1994.

[24] Smolka, G. A Foundation for Higher-order Concurrent Constraint Programming. Tech. rep., Bundesminister fr Forschung und Technologie, 1994.

[25] XXIII-Conferencia Anual del CLEI, Valparaíso(Chile). *$\pi^+ - Calculus$: An extension of the $\pi - Calculus$ to handle Constraints* (1997).

Semiring-based Fuzzy Constraints in Concurrent Constraint Programming

Alberto Delgado, Carlos Olarte, Jorge A. Pérez and Camilo Rueda
Pontificia Universidad Javeriana, Facultad de Ingeniería
Cali, Colombia
albertod@puj.edu.co, {caolarte, japerez, crueda}@atlas.puj.edu.co

Abstract

Several real-life problems have been successfully modeled and solved by using constraint programming (CP). Nevertheless, existing classical (hard) constraints can not express preferences, priorities or other *soft* criteria in a natural way. Since these criteria often occur in many scenarios, finding techniques and tools for appropriately including them in constraint programs is crucial. This paper describes an implementation of a *soft constraints* module for Mozart, a concurrent constraint programming language. The module, based on the semiring formalism for soft constraints, provides an intuitive set of constraints for solving *fuzzy* constraint satisfaction problems and is fully orthogonal to Mozart's implementation. We modify the concept of *constraint* proposed in the semiring formalism in order to define a more intuitive notion. The new concept provides straightforward user control and is suitable for efficient implementation. We present a set of intuitive examples showing the advantages of our module in different contexts. Some issues regarding the integration of soft constraints in existing applications are also discussed.

Keywords: Programming Languages, Constraint Solving, Soft Constraints, Mozart.

1 Introduction

Constraint programming (CP) has been extensively used to model and solve problems in a wide variety of fields, including planning, scheduling, combinatorial optimization and many others. In the last decade CP has experienced considerable advances both in the underlying theoretical basis and in its techniques. However, the CP model is still very limited for expressing criteria such as preferences or priorities, which often occur in many real-world situations. Finding appropriate techniques for handling these kind of criteria is thus fundamental for tackling a significant group of problems.

Introducing the criteria mentioned above within the constraint model has been attempted both from theoretical and practical perspectives. Formalisms for representing "softness" in constraint modeling have mainly been proposed within the framework of constraint satisfaction problems (CSP). Two salient proposals are *valued constraint satisfaction problems* (VCSP [13]) and *semiring-based constraints* (SCSP, [2]).

In this paper we are interested in the appraisal of the SCSP formalism in the context of concurrent constraint programming (CCP). In particular, we are interested in implementing tools supporting use of SCSP techniques in practical situations. We thus extend Mozart [17], a CCP language, to give it a coherent set of SCSP features. Our extension offers an integrated view of hard and soft constraints thus allowing users of traditional CCP systems to easily construct SCSP models. Each constraint needs two new notions: a *consistency function* expressing how far a tuple of values is from any other satisfying the constraint, and a *penalizing factor* representing a unit for measuring the "cost" of accepting an inconsistent tuple. Tuples are then valued according to these notions, also taking into account a user supplied *cut level*. Such a level is global for the whole problem and determines if a tuple should be accepted or not.

The proposed implementation is fully orthogonal to the Mozart language. This is achieved by taking advantage of the flexibility of the language for including new constraint systems by implementing suitable *propagators* (discussed further below). This strategy brings a number of advantages. First, efficient constraint handling can easily be implemented by using existing libraries for building propagators. Second, the module is itself extensible thus leaving room to add new soft constraints. Third, a soft constraint can seamlessly

coexist with the efficient built-in finite domain constraints of the language. The last feature is particularly important since it makes the module readily available to the community of constraint applications developers in Mozart. We believe our work contributes in this way to fill in the gap between theoretical and practical work on soft constraints systems.

The rest of this paper is organized as follows. The main theoretical results of the semiring-based framework for soft constraints are summarized in Section 2. Main ideas behind constraint programming in Mozart are also presented there. The proposed soft constraints module for Mozart, implementing procedures for solving *Fuzzy* CSPs, is introduced in Section 3. Some formal definitions, as well as a full description of available operations are described. Examples showing applications of the module are presented in Section 4. A short account of related work is presented in section 5. Some final remarks are included in section 6.

2 Preliminaries

2.1 Semiring-Based Constraints

Here we briefly summarize the most important definitions and properties of the semiring formalism for soft constraints. A more complete description can be found in [2].

A *semiring* is a tuple $(A, +, \times, \mathbf{0}, \mathbf{1})$ where A is a set and $\mathbf{0}, \mathbf{1} \in A$. $+$, the *additive operator* is closed, commutative and associative. Moreover, its unit element is $\mathbf{0}$. $\times$, the *multiplicative operator*, is a closed, associative operation, such that $\mathbf{1}$ is its unit element and $a \times \mathbf{0} = \mathbf{0} = \mathbf{0} \times a$ holds. In addition, $\times$ distributes over $+$. A *c-semiring* is a semiring with some additional properties: $\times$ is commutative, $+$ is idempotent, and $\mathbf{1}$ is its absorbing element. The idempotency of $+$ is needed in order to define a partial ordering $\leq_S$ over the set A, which serves to compare different elements of the semiring. Such a partial order is defined as follows: $a \leq_S b$ iff $a + b = b$.

A *constraint system* is a tuple $CS = \langle S, D, V \rangle$, where S is a semiring, D is a finite set and V is an ordered set of variables. Given a constraint system $CS = \langle S, D, V \rangle$, where $S = (A, +, \times, 0, 1)$, a *constraint* over CS is a pair $\langle def, con \rangle$, where $con \subseteq V$ is called the *type* of the constraint, and $def : D^{|con|} \rightarrow A$ is called the *value* of the constraint. In this way, a *soft constraint satisfaction problem* (SCSP) P over CS is defined as a pair $P = \langle C, con \rangle$, where C is a set of constraints over CS and con is a subset of V.

Consider any tuple of values t and two sets of variables I and I', with $I' \subseteq I$. $t \downarrow^I_{I'}$ denotes the tuple projection of t w.r.t. the variables in I'. Let $c_1 = \langle def_1, con_1 \rangle$ and $c_2 = \langle def_2, con_2 \rangle$ be two constraints over CS. Then, its *combination* $c_1 \otimes c_2$, is the constraint $c' = \langle def', con' \rangle$, where $con' = con_1 \cup con_2$ and $def'(t) = def_1(t \downarrow^{con'}_{con_1}) \times def_2(t \downarrow^{con'}_{con_2})$. Moreover, given the constraint $c = \langle def, con \rangle$ and a subset w of con, the *projection* of w over c, written $c \Downarrow_w$ is the constraint $\langle def^*, w \rangle$, where $def^*(t^*) = \sum_{\{t | t \downarrow^{con}_w = t^*\}} def(t)$.

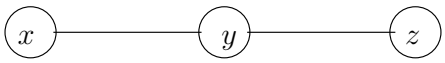
Given an SCSP $P = \langle C, con \rangle$ over a constraint system CS, the *solution* of P is a constraint defined as $Sol(P) = (\bigotimes C) \Downarrow_{con}$ where $\bigotimes C$ is the extension of $\times$ to a set of constraints C. Moreover, an *optimal solution* is a pair $\langle t, v \rangle$ such that $def(t) = v$, and there is no t' such that $v < def(t')$. Sometimes it is enough to know the best value associated with the tuples of a solution. This is called the *best level of consistency:* Given an SCSP $P = \langle C, con \rangle$, the best level of consistency for P is defined as $blevel(P) = (\bigotimes C) \Downarrow_{\emptyset}$. P is said to be consistent if $0 <_S blevel$. In the case where $blevel = \alpha$, P is said to be α-consistent.

In this work we are interested in solving *fuzzy CSPs*. These kind of problems are represented in the semiring-based formalism by the following c-semiring:

$$S_{FuzzyCSP} = \langle \{x \mid x \in [0, 1]\}, \texttt{max}, \texttt{min}, 0, 1 \rangle.$$

Informally speaking, in a Fuzzy CSP each tuple is associated with a real value between 0 (the worst value, the tuple is not allowed) and 1 (the best one, the tuple is allowed). Hence, the goal is to find a set of tuples of values (for all variables in *con*) with maximal valuation. The value of a n-tuple is obtained by *minimizing* the values of its sub-tuples. Alternatively, in a Fuzzy CSP one tries to maximize the value of the least preferred tuple.

Example 1 *Consider the SCSP depicted in Figure 1. It is composed of three finite-domain variables (i.e. x, y, z, represented by nodes) and two constraints (represented by the tuples under the edges between nodes). The domain of the variables is $dom(x) = \{1, 2, 3, 5\}$, $dom(y) = \{1, 2, 4, 5\}$ and $dom(z) = \{2, 3, 4, 5\}$, respectively. The semiring value associated with each tuple is shown next to it. The problem has four solutions:*



$\langle 1,2\rangle$	0.3	$\langle 2,2\rangle$	0.35
$\langle 2,4\rangle$	0.7	$\langle 4,3\rangle$	0.6
$\langle 3,5\rangle$	0.4	$\langle 5,4\rangle$	0.7
$\langle 5,1\rangle$	1.0	$\langle 1,5\rangle$	0.85

Figure 1: A SCSP with two fuzzy constraints over three variables

- $\langle 1,2,2\rangle$ *valued with 0.3*
- $\langle 2,4,3\rangle$ *valued with 0.6*
- $\langle 3,5,4\rangle$ *valued with 0.4*
- $\langle 5,1,3\rangle$ *valued with 0.85*

Note how the valuation for each solution was obtained by minimizing the valuation of its two sub-tuples. For instance, the valuation for $\langle 3,5,4\rangle$ *is equal to* $\texttt{min}(0.4, 0.7)$*, the combination of the values associated with sub-tuples* $\langle 3,5\rangle$ *and* $\langle 5,4\rangle$*. Consequently, the optimal solution for the problem is* $\langle 5,1,3\rangle$*.*

2.2 Constraint Programming in Mozart

Mozart[1] is a concurrent constraint programming language that provides several functionalities, including support for distributed programming, constraint solving, as well as supporting tools for programming. Many real-life problems have been successfully solved with Mozart (see for instance [5, 6]). It also provides efficient built-in constraints over finite domains of integers as well as convenient mechanisms for creating suitable propagators and new constraint systems [11]. Next we provide a concise introduction to Mozart [14, 10, 15].

Mozart considers basic and non-basic constraints. A *basic constraint* is a logic formula interpreted in some first-order structure. These are chosen so that entailment can be efficiently decided. *Non-basic constraints* are relations built from combination of basic constraints. Basic constraints are kept in a monotonic *store*. Non-basic constraints are enforced by *propagators* [15]. A propagator is a computational agent encapsulating a filter function that deduces consequences (i.e. new basic constraints) of the non-basic constraint. A propagator for a constraint c ceases to exist if c is entailed by the current store or if the conjunction of the current store and c is unsatisfiable. In that case, the propagator for c is said to be *disentailed*, since $\neg c$ is entailed by the current store [10]. Typically, propagators share variables. This causes propagators to trigger each other by writing new basic constraints to the store. This continues until a propagation fixed-point is reached [10]. The order in which the propagators add information to the store does not matter.

Computations in Mozart take place in *computation spaces*. A computation space consists of a set of propagators connected to a store. A space S is said to be *stable*, if no further propagation in S is possible. A stable space S is said to be *failed*, if S contains a propagator that disentails some constraint. A stable space S is *solved*, if S contains no propagators [14]. Moreover, a variable assignment is called a *solution* of a space if it satisfies the constraints in the store and all constraints enforced by propagators.

Constraint propagation is not a complete solution method. To achieve completeness, propagation must be combined with a special labelling phase called *distribution*. When distributing a stable space S (not failed nor solved), a new constraint c is chosen, and two new spaces must be solved: $S \wedge c$ and $S \wedge \neg c$. It is important to choose c such that both new spaces trigger further constraint propagation. By proceeding in this way we obtain a *search tree*, where each node corresponds to a space and each leaf corresponds to a space that is either solved or failed. Since the alternatives depend on variables of the problem, a finite search tree can be assumed [15].

A *distributor* is a procedure implementing a distribution strategy. Usually, a distribution strategy is defined on a sequence $x_1, \ldots x_n$ of variables. When a distribution step is necessary, the strategy selects a yet to be determined variable in the sequence and distributes on this variable (that is, imposes a constraint over

[1] `www.mozart-oz.org`

the selected variable). There are several standard possibilities to distribute over a variable x. For instance, a *naive* distribution strategy will select the leftmost undetermined variable in the sequence, while a *first-fail* distribution strategy will select the leftmost undetermined variable with the smallest domain [15].

3 Soft Constraints in Mozart: A propagator approach

Semiring-based constraints have shown a great potential to express and to reason about a variety of problems in constraint programming. However, from a constraint programmer perspective there is no clear idea about how to define a constraint using the semiring formalism. It would be desirable to have a set of efficient and well-defined soft constraints such as those provided for classical constraints in several constraint programming languages. In this sense, a related requirement is that such a set of soft constraints should be fully compatible with the usual classical constraints. This is justified by the fact that most problems are defined with both soft and hard constraints. Our work is aimed at satisfying these two requirements. It brings together semiring concepts into Mozart's constraint propagation model in order to provide a solid implementation of Fuzzy CSPs.

Our module for soft constraints in Mozart offers a rich set of soft propagators as well as provides a clean interface to define new ones. Since both soft and classical constraints follow the same implementation patterns, it is possible for them to interact and coexist transparently both to the user and to the underlying propagation engine. From a programmer point of view, our module offers the flexibility of providing two ways to define soft conditions, namely a *cut level* for establishing a minimal level of satisfaction for the whole problem and a *penalization factor* representing the impact on the overall valuation when a given constraint is violated. We believe that these two levels of "softness" (one global level for the problem, another local for the constraints) constitute a sufficient basis for expressing behavior of soft constraints and allow us to define soft propagators conveniently.

In what follows we precise the above notions, and present a set of soft constraints for Mozart based on them.

Definition 1 (Cut Level) *Given a semiring $S = \langle A, \times, +, \mathbf{0}, \mathbf{1}\rangle$ and a SCSP P, the* cut level $\tau >_s \mathbf{0}$ *is an element in A representing the minimal level of consistency accepted by the user. P is said to be consistent iff $blevel(P) \geq_s \tau$.*

As mentioned in the previous section, a problem P is said to be inconsistent if it is α-consistent with $\alpha = \mathbf{0}$. Nevertheless, this notion of inconsistency is not realistic in practice as some solutions may have valuations greater than $\mathbf{0}$ and, at the same time, be meaningless to the user. The cut level notion makes precise the concept of *useful solution*. In this way, a user tolerance w.r.t. the valuation of a solution can be captured in a more direct way. Moreover, as we will see later, this value can play an active role when pruning the domains of the variables.

Although the notion of constraint is nicely expressed in the semiring formalism by means of the def function, when writing a constraint program it is not natural to think of a soft constraint as a valuation function. For a user, it is more practical to rely on predefined procedures enforcing the constraint instead of *enumerating* the possible values for the Cartesian product of the domain of the variables. Therefore, such procedures must provide mechanisms for handling values associated with tuples efficiently. Such a mechanism can be understood as two components. The first one obtains a *quantitative reference* between a given tuple and a tuple regarded as correct, according to the semantics of the constraint to be defined. Intuitively, this reference (i.e. a natural number) represents a *distance* or degree of correctness between both tuples. The second component translates this distance into a value of the semiring, thus completing the required association for the tuples. Intuitions underlying these two components can be formalized as follows.

Definition 2 (Consistency Function) *Let c and $\sigma_c : D^{con} \rightarrow \mathcal{N}$ be a hard constraint and a function associated with it, respectively. For a tuple $t \in D^{con}$, $\sigma_c(t)$ constitutes a quantitative measure of the consistency of t w.r.t. any tuple consistent with c.*

It is worth noticing that this consistency function is completely independent from the used semiring. The notion of consistency (or correctness) depends only on the semantics of the constraint c. It may differ according to criteria such as efficiency and desired level of detail. Take for instance the `distinct` constraint,

that states that all elements in a sequence of integers must be pairwise distinct [16]. One possible definition of the consistency function for this constraint is the following:

$$\sigma_{\texttt{distinct}}(t) = nrepeat(t)$$

where $nrepeat(t)$ is a function returning the number of repeated elements in a sequence of integers t. For example, $nrepeat([1,1,2]) = 1$ while $nrepeat([1,4,3,7]) = 0$. Nevertheless, one can think of other consistency functions for the `distinct` constraint (see for instance those proposed in [12]). Under this scheme, an appropriate consistency function is one that faithfully describes the soft features related to the constraint while keeping a reasonable complexity.

In order to complete the association of tuples and semiring values, a relationship between distance values (given by the consistency function) and semiring elements must be defined. In principle, there are no restrictions on the nature of such a relationship, so it is possible to provide a definition that enhances the notion of *softness* given by the consistency function. In particular, this relationship can be based on a *factor* attached to each constraint that have incidence on the association of tuples and semiring values:

Definition 3 (Valuation Function) *Assume a consistency function σ_c for a hard constraint c. A* valuation function *$\rho_\beta : \mathcal{N} \to A$ associates every value returned by σ_c with an element in A, being $\beta \in A$ a parameter in such association. β is said to be the* penalization factor *associated with c.*

Summing up, every soft constraint c depends on two complementary functions to associate semiring values and tuples: σ_c y ρ_β. These two notions are captured in the following definition:

Definition 4 (Penalization System) *Given a hard constraint c, a penalization system $\langle \sigma_c, \rho_\beta \rangle$ for it is composed of a consistency function σ_c and a valuation function ρ_β.*

With the previous definitions it is then possible to redefine the notion of constraint in the semiring formalism:

Definition 5 (Soft Constraint) *Let c a hard constraint over a set of constraints con, a constraint system $CS = \langle S, D, V \rangle$ (with $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$) and a penalization system $\langle \sigma_c, \rho_\beta \rangle$ for c. A soft constraint based on c is a pair $\langle def_{\langle \sigma_c, \rho_\beta \rangle}, con \rangle$ where:*

1. *$con \subseteq V$, is called the* type *of the constraint;*
2. *$def_{\langle \sigma_c, \rho_\beta \rangle} : D^{|con|} \to A$, such that $def_{\langle \sigma_c, \rho_\beta \rangle}(t) = \rho_\beta(\sigma_c(t))$, for each tuple $t \in D^{|con|}$.*

A fundamental aspect of this notion is that is completely orthogonal with respect to the original definitions of constraint and soft constraint satisfaction problem: our definition makes the association between tuples and semiring values more precise. More importantly, this additional precision does not imply a loss of generality for the semiring formalism. On the contrary, a wide range of possibilities can be envisioned for defining soft constraint as our proposal can be easily parameterized. The following example illustrates this argument.

Example 2 (A soft constraint) *Consider the hard constraint $X < Y$. A soft version of this constraint for the fuzzy semiring can be defined as $c = \langle \{X, Y\}, def_{\langle \sigma_{lt}, \rho_{0.2} \rangle} \rangle$, where:*

- $\sigma_{\texttt{lt}}(i,j) = \texttt{max}(i - j + 1.0\ ,\ 0.0), \quad \forall_{\langle i,j \rangle} \in dom(X) \times dom(Y)$
- $\rho_{0.2}(x) = 1 - (x * 0.2)$.

Informally, consistency function $\sigma_{\texttt{lt}}$ returns the number of "violation units" of a tuple w.r.t. c. Note that when c is completely satisfied $\sigma_{\texttt{lt}}$ will return 0. Function $\rho_{0.2}$ maps those units to the context of the real numbers between 0 and 1. According to this, semiring values associated with tuples $\langle 4,4 \rangle$ and $\langle 5,4 \rangle$ are $def(\langle 4,4 \rangle) = 1.0 - 0.2 \times (1.0) = 0.8$ and $def((\langle 5,4 \rangle) = 0.6$, respectively.

Additional to the definitions given so far, it is interesting to have a concrete idea of the effects a soft constraint has over a particular SCSP P. In some sense, those effects can be understood as the *tolerance* such constraint has w.r.t. a possible inconsistency of a given tuple. This concept of tolerance, that depends on the cut level τ associated with P, is formalized below:

Definition 6 (Tolerance of a Soft Constraint) *Let $P_1 = \langle C, con\rangle$ and τ_1 be a SCSP and its cut level, respectively. For a soft constraint $c_d \in C$, $tol(c_d)$ is a natural number that represents the* tolerance *of c_d w.r.t. P_1. Such tolerance is computed using β, τ_1 and $\mathbf{1}$, and can be seen as the number of allowed violations for c_d.*

The tolerance of a constraint is a reference of the valuations associated with tuples. It constitutes a direct translation of the user's wishes, that are expressed in terms of β and τ. While valuations associated with tuples (computed using $def_{\langle\sigma_{lt},\rho_\beta\rangle}$) can be regarded as generic elements, the tolerance of a constraint is completely problem-dependant.

From previous definitions it is possible to understand the direct role that the cut level τ and the penalization factor β have over valuations associated with tuples. This is a convenient scheme for implementation purposes, since storing and processing valuation-related data is not necessary: the consistency and valuation functions as well as the penalization factor are associated with each constraint, whereas the cut level is unique for the whole constraint problem. Using these two notions it is possible to define the concept of filter function for a soft propagator:

Definition 7 (Filter function) *Given a soft constraint $c_d = \langle def_{\langle\sigma_c,\rho_\beta\rangle}, con\rangle$, a filter function for it (denoted by F_{c_d}) is a narrowing operator [1] that for every variable $x_i \in con$ and every tuple $t \in D^{con}$ removes all values $d_i \in dom(x_i)$ such that $t \downarrow_{x_i} = d_i$, $def_{\langle\sigma_c,\rho_\beta\rangle}(t) <_s \tau$.*

In other words, the task of the filter function of a soft constraint is to *enforce* its tolerance, by discarding all those values in variables' domains that do not respect it. In this way, the tolerance of a user to constraint violations becomes the most important criteria when defining soft propagators: both the number of solutions and the needed resources to find them depend on such tolerance. From a practical point of view this implies that choosing a τ very close to 0 (representing a high tolerance to constraint violations) leads to a bigger search space than the one associated with a problem with a τ close to 1 (that is, low tolerance to constraint violations).

We are now ready to define a soft propagator:

Definition 8 (Soft Propagator) *Given a soft constraint $c_d = \langle def_{\langle\sigma_{c_d},\rho_\beta\rangle}, con\rangle$, a* soft propagator *for c_d is a computational agent that applies a filter function F_c.*

With this definition the process of adapting the semiring formalism for soft constraints is completed. The notion of propagator as an agent that enforces a constraint by implementing a filter function over variables' domains is common in almost every implementation of constraint-based programming languages.

3.1 Available Operations

The current state of our soft constraints module includes propagators for the following constraints:

Relational constraints of the form {`Soft.relOp D1 D2` β} where `relOp` can be either '$<$', '$>$', '$\geq$', '$\leq$' or '$=$'. These constraints are defined by the following consistency functions (d_i is a value in the domain D_j):

- $\sigma_{less}(\langle d1, d2\rangle) = \mathtt{max}(0, d1 - d2 + 1)$
- $\sigma_{lessEq}(\langle d1, d2\rangle) = \mathtt{max}(0, d1 - d2)$
- $\sigma_{greater}(\langle d1, d2\rangle) = \mathtt{max}(0, d2 - d1 + 1)$
- $\sigma_{greaterEq}(\langle d1, d2\rangle) = \mathtt{max}(0, d2 - d1)$

Arithmetic constraints of the form {`Soft.arithOp D1 D2 D3` β} where `arithOp` is either '$+$', '$\times$', '$-$', '$\div$', '*mod*' or '*pow*'. The consistency function for these constraints is defined by

$$\sigma_{\mathtt{arithOp}}(\langle d1, d2, d3\rangle) = |D3 - (D1\ \mathtt{arithOp}\ D2)|.$$

Global Constraints such as `distinct` where $\sigma_{distinct}(t) = nrepeat(t)$ as defined before. In addition, the soft constraint {`Soft.distance D1 D2 relOp D3` β} is defined with the following consistency function:

$$\sigma_{\mathtt{dist}}(\langle d1, d2, d3\rangle) = \sigma_{relOp}(\langle |d1 - d2|, d3\rangle).$$

Soft constraints with explicit def ***functions*** We can assert constraints $c = \langle def, con \rangle$ by using {`Soft.n-aryPreference LVar`}. In this case, we discard elements with no support w.r.t. τ.

In addition, the module provides search methods `Soft.searchAll` and `Soft.searchOne` for obtaining solutions with valuations.

4 Examples

In this section we describe some examples illustrating the functionality of our implementation as well as several important issues related to it. These include: user commands (mostly constraints) that ease the task of writing soft constraint programs, computational costs of using soft statements, some strategies for introducing soft statements in existing applications and the use of theoretical results to find acceptable values for the cut level. We believe these issues are crucial in order to envision an industrial and commercial use of soft constraint programming.

All tests presented in this section were performed on a PC with a Xeon Processor (2.4 GHz) with 1GB RAM, running Linux Gentoo (Kernel 2.6.5) and Mozart 1.3.1. For the sake of brevity, some technical and/or unnecessary details have been dropped from the example programs.

4.1 A well known example: The zebra puzzle

Our first example deals with solving the well known *Zebra puzzle*. This problem tries to assign a house to five men with different nationalities. There are five consecutive houses in a street for them. The men practice distinct professions, and each of them has a favorite drink and a favorite animal, all of them different. The five houses are painted with different colors. The known facts are described in table 1.

1. *The Englishman lives in a red house.*
2. *The Spaniard owns a dog*
3. *The Japanese is a painter*
4. *The Italian drinks tea*
5. *The Norwegian lives in the first house*
6. *The owner of the green house drinks coffee*
7. *The green house comes after the white one*
8. *The sculptor breeds snails*
9. *The diplomat lives in the yellow house.*
10. *Milk is drunk in the third house.*
11. *The Norwegian's house is next to the blue one.*
12. *The violinist drinks juice.*
13. *The fox is in the house next to that of the doctor.*
14. *The horse is in the house next to that of the diplomat.*
15. *The zebra is in the white house.*
16. *One of the men drinks water.*

Table 1: Known facts in the Zebra puzzle

In [15], this problem is modeled as depicted in figure 2. Using that piece of code, Mozart only returns the solution displayed in table 2. A demanding user could consider this solution as rigid, and may wish other solutions. He/she does not care about violating some of the constraints. Let us follow this idea and consider procedure `Adjacent`. This procedure is used for including facts 11, 13 and 14 into the problem. It consists of a constraint imposing the relationship $|X - Y| = 1$, X and Y representing two houses that must be next to each other. Suppose that, for some reason, the fox and the doctor forget their constraint of living on a different house. That is, we consider fine a solution maintaining that condition but it is also fine if it does not. This new relaxed condition can be easily introduced in the problem by using a new `Adjacent` procedure to assert fact 13:

```
proc {SoftAdjacent X Y}
    {Soft.distance X Y '=' 1 0.1}
end
```

```
proc {Zebra Nb}
    proc {Adjacent X Y}
       {FD.distance X Y '=:' 1}
    end
 in
    %% Nb maps all properties to house numbers
    {FD.record number Properties 1#5 Nb}
    {ForAll Groups Partition}
    Nb.english = Nb.red          Nb.spanish = Nb.dog
    Nb.japanese = Nb.painter     Nb.italian = Nb.tea
    Nb.norwegian = 1             Nb.green = Nb.coffee
    Nb.green >: Nb.white         Nb.sculptor = Nb.snails
    Nb.diplomat = Nb.yellow      Nb.milk = 3
    Nb.violinist = Nb.juice      Nb.zebra = Nb.white
    {Adjacent Nb.norwegian Nb.blue}
    {Adjacent Nb.fox Nb.doctor}
    {Adjacent Nb.horse Nb.diplomat}
    {FD.distribute ff Nb}
end
```

Figure 2: Hard constraint model in Mozart for the Zebra puzzle.

House	1	2	3	4	5
Nationality	Norwegian	Italian	Spanish	English	Japanese
Color	White	Blue	Yellow	Red	Green
Drink	Juice	Tea	Milk	Water	Coffee
Profession	Violinist	Sculptor	Diplomat	Doctor	Painter
Animal	Zebra	Snails	Dog	Horse	Fox

Table 2: Unique solution for the original Zebra problem

Using this new procedure with a cut level $\tau = 0.9$ the number of solutions increased from 1 to 10. One of them is shown in table 3. Although the new solutions do not respect the original set of conditions, they are concrete alternatives to solve the problem. The overhead in execution time caused by the relaxation is negligibly small: while solving the hard problem took 287 milliseconds, solving the soft one took 289 milliseconds. This good performance may be explained by the fact that τ is close to 1; our next example studies the relationship between cut level, number of solutions and overall performance.

Note that expressing soft constraints is simple and intuitive. Most of them keep the same syntax as the corresponding hard constraints. This is specially useful for those users with little knowledge of the soft semiring formalism since to include relaxed statements in Mozart programs may consist only in changing just a few constraints. The issue of which constraints should be changed when relaxing problems is addressed in our next example.

4.2 Integrating Soft Constraints into Existing Applications

Here we present a modification of another known problem in the literature in order to introduce some strategies and ideas to improve the handling of soft constraints.

An over-constrained version of the n-queens problem Consider the n-queens problem (see [15], section 5). The idea is to place n queens in a $n \times n$ board such that no two queens attack each other. Here we deal with an over-constrained version of this problem that tries to place n queens into a $n \times (n-1)$ board. We intend to show how a careful choice of the cut level can help to avoid absurd solutions and, at the same time, to maintain a good overall performance.

This problem is modeled as follows. We assume queens are numbered from 1 to n. The k-th queen is placed in the k-th column. For each queen i, a variable R_i represents the row where the queen should be placed. The domain of these variables is the set of integers between 1 and $n-1$. Two kinds of constraints are asserted: hard constraints preventing diagonal attacks and a soft version of the distinct constraint. Note that this relaxed constraint enables us to obtain solutions.

House	1	2	3	4	5
Nationality	Norwegian	Italian	English	Japanese	Spanish
Color	White	Blue	Red	Green	Yellow
Drink	Juice	Tea	Milk	Coffee	Water
Profession	Violinist	*Doctor*	Sculptor	Painter	Diplomat
Animal	Zebra	*Fox*	Snails	Horse	Dog

Table 3: One possible soft solution (valuated with 0.9) for the Zebra problem. Note that the Doctor and the Fox are in the same house.

```
fun {SoftQueens N}
   proc {$ Row}
      L1N ={MakeTuple c N}                              % (1)
      LM1N={MakeTuple c N}                              % (2)
      {Soft.setClevel 0.9}                              % (3)
   in
      {FD.tuple queens N 1#N-1 Row}                     % (4)
      {For 1 N 1 proc {$ I} L1N.I=I LM1N.I=~I end}      % (5)
      {Soft.distinct Row 0.1}                           % (6)
      {FD.distinctOffset Row LM1N}                      % (7)
      {FD.distinctOffset Row L1N}                       % (8)
      {FD.distribute generic(value:min) Row}            % (9)
   end
end
```

Figure 3: `SoftQueens` procedure

We implemented this problem in Mozart using the `SoftQueens` function in Figure 3. The function takes the number of queens (i.e. n) as a parameter. In the code, some auxiliary n-tuples needed to assert diagonal constraints are created in lines (1) and (2). `Row`, the variable that will contain the solution for the problem, is defined in line (4). This variable is composed of the R_i variables mentioned before. The no-attack constraints are imposed in lines (7) and (8). Finally, a distribution strategy is defined in line (9).

There are only two soft statements in `SoftQueens`. The first one, in line (3), asserts that we are only interested in those solutions above the 90% of consistency (or satisfaction). This guarantees that useless solutions are discarded. In line (6), the soft distinct constraint is imposed with $\beta = 0.1$. Recall the consistency function associated with this constraint returns the number of repeated elements in the given list of variables. Therefore, only those solutions having at most a repeated element will be accepted as solutions. Getting all solutions for $n = 9$, is expressed in Mozart as follows:

```
{Soft.searchAll {SoftQueens 9}}
```

This search process returns 572 solutions valued with 0.9 (see Figure 4). This lead us to analyze a feature of our implementation, namely the strong relationship existing between the level (or degree) of relaxation for the constraints and the number of solutions that can be obtained. This relationship, that can be explained by the growth in the search space induced by the relaxation of a constraint, has an adverse influence in the overall performance of the system. This situation can be seen in table 4, that shows the behavior of our propagators running {`SoftQueens 9`} using different values for the cut level. It is easy to see that as the cut level decreases, the size of the problem (i.e. the number of nodes in the search tree to be explored) dramatically increases. For instance, changing the cut level from 90% to 80% implies a growth of 176% in the size of the search tree. This fact is more dramatic when changing such a level to 70%: the size of the problem increases in 622%. It is important to note that this behaviour may not be the same when searching for just one solution. Moreover, that one solution could be found inmediately.

This example exhibits a clear interaction between hard constrains (those imposing diagonal rules) and soft constraints (`Soft.distinct`). This implies that we are interested in enforcing the diagonal rules while we have less expectations from the distinct constraints. Note that this is a specific understanding of the soft features of the problem. Usually, there are several alternatives when determining which constraints are hard and which are soft. This kind of trade-offs are quite frequent in real-life situations and they certainly contradict the idea that all problems must be formulated using only soft constraints.

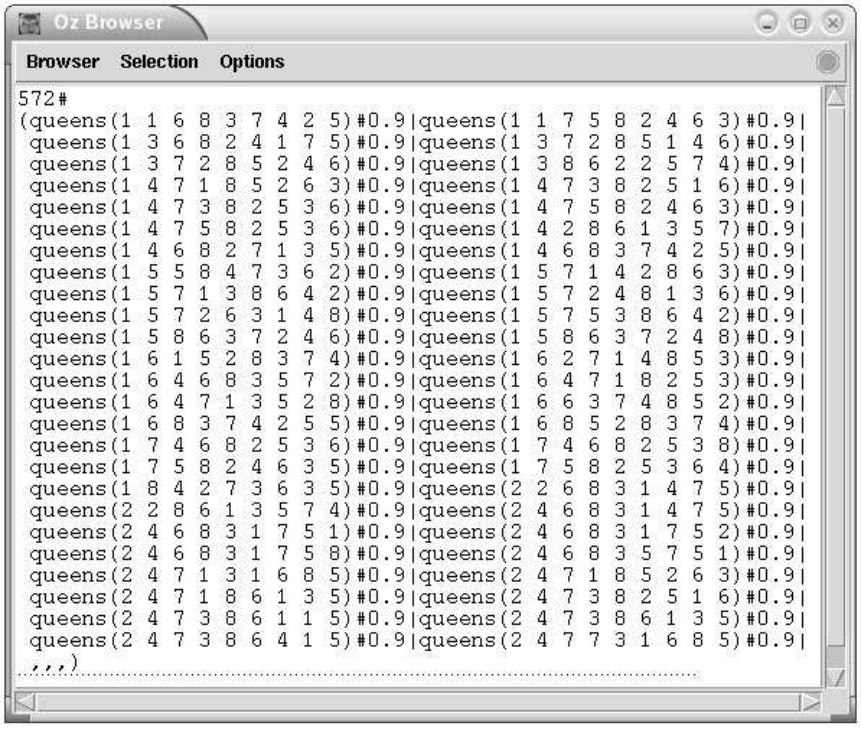


Figure 4: Mozart's *Browser* reporting 572 solutions for the $n \times (n-1)$-queens problem.

Cut Level	Explored Nodes	Failed Nodes	Solutions	Time (s)
0.9	8702	8131	572	1.35
0.8	15341	10372	4970	2.54
0.7	54134	20579	34556	10.44
0.6	58682	7351	51332	12.62

Table 4: A comparison of the performance of {`SoftQueens 9`} for a number of cut levels.

Although the soft queens example is very simple, it shows the importance of a careful choice of soft statements when relaxing a constraint application. We regard the use of soft constraints implementations in real scenarios as centered around two basic factors:

- Modifications needed on existing constraint applications that wish to use soft constraints.
- Agreements regarding the obtained solutions when using a soft constraints implementation.

The first item is related with the cost of introducing soft constraints in an existing application. Although soft constraints allow a more faithful representation for constraint models, stating all or most of the constraints in a problem in terms of soft constraints is computationally harder, because soft propagators perform less pruning action than hard ones. Consider any commercial application: the costs, in time and money, of changing the application are could be prohibitive; performance decrease due to soft constraints could be also significant. This fact was clear in our soft queens example, as a high cost had to be payed to obtain more (albeit worse) solutions.

For this reason, we consider that adding soft constraints in real settings depends on the identification of a specific set of constraints to be relaxed. The penalization factors associated with the violation of these constraints must also be determined. Naturally, such a set must contain those constraints that reflect optional features of the problem. Think of any application in operations research: constraints regarding the number of available resources can be relaxed, since some kind of arrangements are possible in real life. On the contrary, constraints stating mandatory conditions (like business rules), cannot be replaced by their soft

counterpart. Moreover, this replacement (or relaxing) of constraints is related with the second item stated above: the agreement process derived from the approximate solutions obtained by using soft constraints.

By using soft constraints, the programmer must negotiate with the final user those solutions that are good enough w.r.t. the constraints in the problem, but does not hold for all of them. Moreover, such approximate solutions will require additional effort by the user. This implies that the programmer (and the final user) must be willing to deal with not-so-good solutions as a result of the software development process. We believe that either the process of convincing the user of accepting an approximate solution and/or the effort of the user in arranging some conditions in its real setting, will be easier if the relaxed constraints are carefully chosen. This process may also help to avoid "absurd" solutions, useless in practice.

To conclude, it is clear that the use of soft constraints in existing applications can be very useful, but its inclusion must be carefully planned. Since our module for soft constraints in Mozart can be consistently used in conjunction with the efficient, existing hard mechanisms (FD propagators), the main task for the programmer is to select and replace crucial constraints in the problem. This choice will influence the rest of the development process, since approximated solutions can be more easily accepted by the final users of the application if the changes to make are reasonably manageable.

5 Related Work

There exist previous implementations of the semiring-based constraints framework. Most of them are supported by the Constraint Logic Programming (CLP) model. Some instances are `clp(FD,S)` [7], the semiring integration with CHRs [3] and the evaluation approach proposed in [8]. One main drawback of these implementations is that they seem to handle only small problems. Both the CHR module and the `clp(FD,S)` extension provide procedures or predicates that allow to write easy to understand programs.

Two similar notions to our idea of cut level can be found in the literature. The first of them, *α-cuts*, is heavily used in fuzzy set theory [9]. In this context, an alpha-cut of the membership function A (denoted ${}^{\alpha}A$) is the set of all x such that $A(x)$ is greater than or equal to α. Our cut level notion can easily be understood in this context. In our case, we are dealing with the (fuzzy) set $\mathcal{U}$ of useful solutions. Thus, τ works as the α-cut of the membership function of the set $\mathcal{U}$.

Another notion similar to our idea, also called *cut level*, is proposed in [4] as a new primitive included in the cc syntax in order to extend formal foundations of the CCP model with soft constraints constructs. As in our work, the proposed notion of cut level works as a threshold that determines if an agent must suspend, fail or entail. This threshold is included in both *ask* and *tell* rules. It is important to distinguish between the theoretical interests of [4] and the practical ideas behind our work.

6 Concluding Remarks

In this paper we have defined robust mechanisms that orthogonally integrate soft constraints in a CCP setting. This is an important aspect for most combinatorial problems as they usually include soft criteria like preferences and priorities. For doing so, we brought together concepts from the semiring-based constraints formalism for soft constraints and the propagation model of the Mozart language. In this way, a transparent interaction between hard and soft constraints is achieved. This interaction, occurring both from the programmer point of view and from the propagation engine of Mozart, provides users the ability of modeling CP problems with soft constraints in a straightforward way.

In order to relax constraints, two formal concepts were added to the semiring formal framework: a cut level (τ) expressing the global tolerance of the user w.r.t. solutions found by the solver, and a penalization factor (β) that precisely defines how soft a constraint is. A very important property of our implementation is that no semiring valuation data must be stored. Moreover, valuations can be obtained from a function that is wired to the semantics of each constraint. Thus, the formal approach is coupled with a clean and understandable syntax provided by the soft constraints module. This syntax resembles classical constraints available in any constraint programming language.

We also gave several examples using our soft constraints module. We studied the impact of τ on the search space for a relaxed problem and on the number and quality of the obtained solutions. This feature requires that the inclusion of soft constraints in existing applications must be carefully planned as the costs (in time and resources) can increase dramatically because of such an inclusion.

References

[1] F. Benhamou, D. McAllester, and P. Van Hentenryck. Clp(intervals) revisited. In *Proceedings of ILPS 94 - MIT Press*, pages 1–21, 1994.

[2] Stefano Bistarelli. *Semirings for Soft Constraint Solving and Programming*, volume 2962 of *LNCS*. Springer-Verlag, 2004.

[3] Stefano Bistarelli, Thom Fruhwirth, Michael Marte, and Francesca Rossi. Soft constraint propagation and solving in constraint handling rules. In *Proceedings of the Third Workshop on Rule-Based Constraint Reasoning and Programming*, 2001.

[4] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Soft concurrent constraint programming. In *European Symposium on Programming*, volume 2305 of *LNCS*, pages 53–67. Springer, 2002.

[5] Alberto Delgado, Jorge Andrés Pérez, Gustavo Pabón, Rafael Jordan, Juan F. Díaz, and Camilo Rueda. An Interactive Tool for the Controlled Execution of an Automated Timetabling Constraint Engine. In *Multiparadigm Programming in Mozart/Oz*, volume 3389 of *LNCS*. Springer-Verlag, 2005.

[6] Juan F. Diaz, Gustavo Gutierrez, Carlos Olarte, and Camilo Rueda. CRE2: a CP application for reconfiguring a power distribution network for power losses reduction. In *Tenth International Conference on Principles and Practice of Constraint Programming*, volume 3258 of *LNCS*. Springer-Verlag, 2004.

[7] Yan Georget and Philippe Codognet. Compiling semiring-based constraints with clp(fd,s). In *Proc. of CP'98*, volume 1520 of *LNCS*. Springer, 1998.

[8] Jerome Kelleher and Barry O'Sullivan. Evaluation-based semiring meta-constraints. In *Proceedings of MICAI*, volume 2972 of *LNCS*. Springer, April 2004.

[9] George J. Klir and Bo Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, 1995.

[10] Tobias Müller. *Constraint Propagation in Mozart*. PhD thesis, Universitat des Saarlandes, 2001.

[11] Tobias Muller. The Mozart Constraint Extensions Reference. Available at `www.mozart-oz.org`, April 2004.

[12] T. Petit, J.C. Régin, and C. Bessiere. Specific filtering algorithms for over-constrained problems. In *Proc. of CP'2001*, volume 2239 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[13] Thomas Schiex, Hélène Fargier, and Gerard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proc. of IJCAI'95*, Montreal, 1995.

[14] Christian Schulte. *Programming Constraint Services*. PhD thesis, Universitat des Saarlandes, 2001.

[15] Christian Schulte and Gert Smolka. Finite Domain Constraint Programming in Oz - A Tutorial. Available at `www.mozart-oz.org`, April 2004.

[16] Christian Schulte and Gert Smolka. *Finite Domain Constraint Programming in Oz. A Tutorial.*, 2004. Available at `www.mozart-oz.org`.

[17] Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, volume 1000 of *LNCS*, pages 324–343. Berlin, 1995.

Predicción de Pseudonudos en la Estructura Secundaria de ARN vía Gramáticas Estocásticas Independientes del Contexto

Marcos Niño
Universidad de los Andes, Departamento de Ingeniería de Sistemas y Computación,
Bogotá, Colombia
m.nino68@egresados.uniandes.edu.co

y

Rafael García
Universidad de los Andes, Departamento de Ingeniería de Sistemas y Computación,
Bogotá, Colombia
rgarcia@uniandes.edu.co

Abstract

Pseudoknots are a frequent RNA structure that assumes essential roles for varied biocatalyst cell's functions. One of the most challenging fields in bioinformatics is the prediction of this secondary structure based on the base-pair sequence that dictates it. Previously, a model adapted from computational linguistics has been used to predict RNA secondary structure, called Stochastic-Context-Free Grammars (SCFG). However, to this date this approach impose an impractical complexity cost [$O(n^4)$] when they are applied to the prediction of pseudoknots, mainly because a context-sensitive grammar is formally required to analyze them. Other hybrids (energy maximization) approaches give a complexity of $O(n^3)$ in the best case scenario, besides having several restrictions in the sequence maximum length for practical analysis.
Here we introduce o novel algorithm, based on pattern matching techniques, that uses a sequential approximation strategy to solve the original problem. This algorithm not only reduces the complexity to $O(n^2 \log n)$, but also widens the sequence maximum length, as well as the capacity of analyzing various pseudoknots simultaneously.

Keywords: pseudoknots, Stochastic Context-Free Grammars (SCFG), secondary structure prediction, RNA, dynamic programming.

Resumen

Los pseudonudos son estructuras frecuentes del ARN que toman roles esenciales para diversas funciones biocatalizadoras de la célula. Uno de los campos más desafiantes de la bioinformática es la predicción de dicha estructura molecular a partir de la secuencia de bases que la genera. Previamente se ha utilizado un modelo adaptado de la lingüística computacional para predecir la estructura secundaria del ARN, las Gramáticas Estocásticas Independientes del Contexto (SCFG). Sin embargo, hasta el momento esta aproximación impone un costo prohibitivo en complejidad cuando se intentan predecir pseudonudos [$O(n^4)$], ya que formalmente se requeriría una gramática sensitiva de contexto para tal fin. Otras aproximaciones híbridas arrojan complejidades en el mejor de los casos de $O(n^3)$, además de ser soluciones prácticas sólo para moléculas pequeñas de ARN.
Aquí introducimos un algoritmo novedoso, que, inspirado en técnicas de reconocimiento de patrones, utiliza una estrategia de aproximaciones sucesivas para solucionar el problema. Este algoritmo no sólo reduce la complejidad a $O(n^2 \log n)$ sino que aumenta tanto el tamaño de secuencia analizable como la capacidad de analizar varios pseudonudos simultáneamente.

Palabras claves: pseudonudos, Gramáticas Estocásticas Independientes del Contexto (SCFG), predicción de estructura secundaria, RNA, programación dinámica.

1. Introducción

La bioinformática es una ciencia aplicada donde se usa teoría y tecnología matemática/computacional para procesar, relacionar y derivar predicciones e inferencias a partir de datos obtenidos en biología molecular. Esta ciencia pretende sobre todo entender y analizar el flujo y el control de información presente en los diferentes organismos. Aquí interactúan sinérgicamente las ciencias de la computación, las matemáticas y la biología, con todas las riquezas y limitantes que pueda aportar cada una de ellas.

Uno de los ámbitos fundamentales de la bioinformática es el análisis de secuencias biológicas, ámbito que asume la representación de las moléculas esenciales para la vida, (ADN, ARN y Proteínas) como cadenas de residuos que, al poder ser representados con símbolos en un alfabeto particular, pueden ser sujetos a análisis gramaticales para encontrar relaciones entre dichos residuos. Estas relaciones determinan, de acuerdo con el dogma central[1] de la biología molecular, las propiedades biológicas de dichas moléculas [20] [17]. Esta representación secuencial de una molécula se conoce como su estructura primaria.

Al igual que muchas proteínas, las moléculas de ARN presentan apareamientos distantes dentro de una secuencia. Estos acoplamientos (estructura secundaria) se manifiestan con enlaces entre nucleótidos ubicados lejana e indistintamente en la secuencia, lo que produce un doblamiento estructural, conocido como pseudonudo[2]. Desde su descubrimiento [16], estos pseudonudos han sido de particular atención, puesto que confieren una estructura tridimensional a la molécula que determinará en muchos casos su función biológica particular[3].

Si bien determinar esta estructura molecular es de vital importancia, es particularmente difícil, dispendioso y costoso obtener datos estructurales por espectrometría y cristalografía de ARN, por lo que su predicción a priori con base en la secuencia de residuos es un tema esencial en la bioinformática [19]. A continuación describimos brevemente las aproximaciones más utilizadas para tal fin.

Molecularmente es plausible que el doblamiento de ARN sea dictado más por características biofísicas que por el simple conteo y relación de pares de bases. El Algoritmo de minimización de Zuker [26] asume que la correcta configuración estructural es aquella que presenta la energía de equilibrio ΔG^o más baja. Esta predicción se afina combinándola con datos experimentales e información estadística y termodinámica [23]. Sin embargo, dado que el modelo de energía exacta asociada con los pseudonudos aún no está disponible, los modelos existentes utilizan aproximaciones de energía estructural secundaria básica con relativo éxito [12] [9]. Estas aproximaciones no determinan una solución óptima y casi siempre no pueden enunciar qué tan alejada está la solución obtenida de la óptima [19]. Por esto, en la actualidad no existe un algoritmo que pueda predecir satisfactoriamente las clases de pseudonudos; de hecho, los aportes más significativos siguen presentando complejidades que, en la práctica, los hacen inutilizables[4].

Existen varias aproximaciones lingüísticas a los pseudonudos (variaciones de Gramáticas Estocásticas Independientes del Contexto (SCFG[5])), que también buscan predecir sus estructuras y encontrarlas en las bases de datos. Por su misma definición, los pseudonudos son descritos por un lenguaje de copiado. Por esta razón, formalmente se necesitaría una gramática sensitiva de contexto para analizarlos, los correspondientes algoritmos serían de complejidad alta puesto que se trata, en general, de un problema NP-Completo [6]. Para evitar este problema, los métodos lingüísticos intentan basarse en lo siguiente: intersecciones de SCFG asociadas [2]; símbolos no terminales especiales y producciones específicas para cada problema [18]; o gramáticas paralelas que se comunican entre sí [4]. Sin embargo, sus implementaciones requieren tiempos computacionalmente prohibitivos de hasta $O(n^6)$ y todavía no son lo suficientemente eficaces ni fidedignos [19].

En este artículo proponemos un algoritmo alterno para solucionar los anteriores problemas clásicos. La propuesta consiste en alterar un algoritmo eficaz para el análisis de estructura secundaria llamado Modelos de Covariación (CM[6]) [7], de tal manera que sea capaz de analizar estructuras de pseudonudos. Esto se logra desagregando la predicción de pseudonudos en dos predicciones de estructuras secundarias consecutivas y relacionadas, repartiendo así el análisis en

---

[1] El dogma central declara lo siguiente:
- a La secuencia del ADN determina la secuencia de la proteína.
- b. La secuencia de la proteína determina la estructura proteica.
- c. La estructura proteica determina la función proteica.

[2] Más formalmente, un pseudonudo ocurre cuando una región de una cinta de ARN perteneciente a un bucle forma uno o más pares de bases Watson-Crick con una secuencia complementaria por fuera de dicho bucle. Existen dos tipos de pseudonudos, los H-pseudonudos (simples) y los P-pseudonudos (recursivos)

[3] Los pseudonudos existen virtualmente en todas las clases de ARN (ARNm, ARNt, lsuARN, ssuARN, 7-SL ARN, U2-snARN, el grupo I de intrones, ARN viral) y posee varias funciones biológicas esenciales (ver [12], [8], [15] o [5]).

[4] Por ejemplo, el algoritmo de Rivas y Eddy [17] presenta complejidad temporal $O(n^6)$ y espacial $O(n^4)$, además de un reducido espectro (secuencias muy pequeñas, máximo 150 bases) [24]; en la propuesta presentada en [14] la complejidad temporal es $O(n^4)$ y la espacial es $O(n^3)$; y únicamente detectan los pseudonudos de tipo H (los más sencillos); o un algoritmo comparativo que también utilice una heurística de dividir y conquistar, tenga complejidad temporal $O(n^2)$ y permita todo tipo de pseudonudos. Sin embargo, estas aproximaciones son hechas *ad hoc* y requieren intervención manual y experta [24]. Por último, en el algoritmo de ILM, se combina la aproximación termodinámica con la comparativa, que es más sensible y específico que sus antecesores. Sin embargo, se sigue sacrificando optimización en su resultado final [19].

[5] Del inglés *Stochastic Context-Free Grammar*.

[6] Del inglés *Covariation Models.*

dos iteraciones. De esta manera se evita el análisis simultáneo y su correspondiente complejidad y costo computacional. Por este método se predice la estructura en $O(n^2 \log n)$, mejorando así las complejidades de los algoritmos desarrollados hasta el momento y ampliando el tamaño máximo posible de la secuencia a analizar.
Lo anterior se inspira en la teoría de la información [21] y su tratamiento de la entropía, donde se afirma que la información no siempre se encuentra distribuida uniformemente a lo largo del mensaje (secuencia) sino que existen símbolos o regiones que contienen más información que otra. De esta manera, conservando o analizando estas regiones "especiales" se puede obtener una mejor comprensión de todo el mensaje, mensaje que la mayoría de las veces, como lo es en el caso de la bioinformática desde el punto de vista del investigador, es un mensaje críptico que encierra muchos interrogantes.
Este artículo se estructura de la siguiente manera: en la segunda sección se introducen los conceptos básicos para el uso de gramáticas que describen secuencias de ARN y los conceptos básicos asociados a las Gramáticas Estocásticas Independientes del Contexto (SCFG). En la tercera sección se exponen los elementos constitutivos de los Modelos de Covariación, así como una propuesta innovadora para implementar la predicción de pseudonudos en la estructura secundaria como una extensión a tales modelos. En la cuarta sección se presenta una evaluación experimental de la propuesta, y en la última las conclusiones y propuestas de trabajo futuro.

2. Predicción de Estructura Secundaria Vía Gramáticas Independientes del Contexto (SCFG).

2.1 Conceptos básicos

Una de las técnicas más prometedoras en bioinformática es el análisis de gramáticas estocásticas, puesto que permiten generar patrones de secuencias de manera natural, además de tener un mayor rango de acción que otras arquitecturas [20].
Las gramáticas estocásticas tienen su origen en gramáticas formales que fueron desarrolladas como un modelo para analizar los lenguajes naturales; de hecho, éstas fueron desarrolladas al mismo tiempo que el modelo en que la doble hélice fue descrita por Watson y Crick [1]. Las gramáticas son herramientas útiles para modelar cadenas de caracteres, por lo que, en cierta medida, también lo son para modelar secuencias de biología molecular [17] [1] [3]. Muchos problemas de bioinformática pueden ser reformulados en términos de lenguajes formales, logrando producir la correspondiente gramática a partir de los datos disponibles [1].
Entre las varias utilidades que aportan las gramáticas, la principal es la capacidad de decidir, mediante derivaciones, si una cadena es sintácticamente correcta; es decir, si pertenece a un lenguaje determinado. Una derivación puede ser representada como una estructura tipo árbol, conocida como árbol de derivación. Este árbol refleja la estructura sintáctica de una secuencia. Es posible que para una secuencia particular exista más de un árbol de derivación, en este caso se dice que la gramática es ambigua. En gramáticas ambiguas se aumenta la complejidad de derivación dado que el número de árboles posibles crece exponencialmente con respecto a la longitud de la secuencia a derivar [1]. Para una revisión completa de los conceptos básicos de la teoría de lenguajes formales, incluido el estudio de gramáticas, se recomienda la lectura de [10] ó [13].

2.2 Gramáticas Estocásticas Libres del Contexto (SCFG)

Existen muchos ejemplos de palíndromos presentes en el ARN/ADN. Los palíndromos biológicos tienen una diferente connotación a la usual, pues sus letras no son idénticas sino base-complementarias[7], comenzando desde los dos extremos. Esto es, de la misma manera que usualmente se entiende que *reconocer* es un palíndromo dado que la secuencia es igual a la secuencia invertida, se entiende que AGAUUUCGAAUCU es un palíndromo biológico dado que al ser leído de derecha a izquierda se lee una secuencia de bases complementarias a las de la secuencia original. Es decir, dada la complementariedad de las estructuras de doble hélice del ADN, cada mitad de un palíndromo en un lado tiene su imagen especular en la otra cinta. Por esta razón, si una cadena palíndromo se lee de izquierda derecha en un lado, de derecha a izquierda se leerá la misma cadena en la cinta opuesta[8] [1].
Estos apareamientos distantes hacen que las herramientas computacionales de análisis de secuencias, comúnmente utilizadas para proteínas, no pueda ser trasferida al análisis de estructura secundaria de ARN. Esto se debe a que frecuentemente tienen una estructura anidada que no puede ser modelada eficientemente usando correspondencias

[7] En el ARN las bases **A** (Adenina) **C** (Citosina) forman enlaces químicos apareándose, de manera complementaria, con las bases **U** (Uracilo) y **G** (Guanina), respectivamente.
[8] Esta ambigüedad estructural confiere diferentes roles al mismo elemento de ARN, lo que, evolutivamente hablando, es una gran ventaja competitiva para el organismo [22]. Por ejemplo, la estructura secundaria asociada a un palíndromo recursivo es un tallo con otro tallo surgiendo de su lado. Varias estructuras de palíndromos recursivos generan estructuras de tallos ramificados, conocidas como estructuras secundarias ortodoxas, por ejemplo, la estructura de trébol en ARN de transferencia, fundamental en la traducción del ARN mensajero a Proteínas[1].

clásicas de secuencias (Redes Neuronales, lenguajes regulares o Modelos Escondidos de Markov (HMM[9])). Por esta razón, una SCFG puede ser una mejor aproximación [6].
Una SCFG se construye mediante la adición de una estructura probabilística a las reglas de producción de una gramática dada [1]. En otras palabras, a cada regla de producción $\varphi \rightarrow \delta$ se le asigna una probabilidad $P(\varphi \rightarrow \delta)$, la probabilidad de utilizar la regla $\varphi \rightarrow \delta$ en una derivación, cumpliéndose así el segundo axioma de probabilidad, $\Sigma(P_\delta (\varphi \rightarrow \delta))= 1$.
Una gramática estocástica está entonces caracterizada por un conjunto de modelos probabilísticos que generan el lenguaje correspondiente.
De esta manera, para medir la probabilidad P(O|w) [10] de la derivación estocástica de una secuencia O acorde con una gramática M = M(w) con parámetros w, basta calcular:

$$P(O|w) = \Sigma P (S \rightarrow_\pi O|w)$$

donde π es la secuencia de derivaciones necesarias para producir O .

3. Modificación a los Modelos de Covariación como Aproximación a la Predicción de Estructura Secundaria

3.1 Construcción de Modelos de Covariación

Usando una aproximación heredada de los HMM, los Modelos de Covariación (CM) especifican una arquitectura de SCFG adecuada para modelar estructuras de ARN secundarias de consenso. Una estructura de consenso son patrones repetitivos en una familia de ARN que comparten varios *motifs* estructurales entre ellos[11] [6]. Este método se inspira en el algoritmo CYK de programación dinámica, que ha sido el estándar para analizar alineamientos de SCFG. En síntesis, dicho algoritmo encuentra un árbol de derivación óptimo para un modelo parametrizado en una secuencia, y es una optimización del algoritmo *Inside / Outside* [11]. A su vez, este algoritmo calcula el mejor puntaje asignado a una secuencia dada, a partir de las varias alternativas con las cuales una SCFG puede llegar a generarla.
Para describir este alineamiento múltiple entre secuencias de ARN homólogas con una SCFG, son necesarios varios tipos de no terminales para modelar las diferentes estructuras conocidas.
Los no terminales de la gramática presentes en los CM y su semántica son (ver Figura 1) [6]:

- **P**: Las columnas apareadas en puentes de Watson y Crick (enlaces A-U C-G) se describen mediante un no-terminal que emite *un apareamiento de bases*.
- **L**: Las columnas que no están apareadas (solitarias en el espacio), son descritas por un no-terminal que emite *hacia la izquierda* (dirección 5´→ 3´)[12] en la medida de lo posible; es decir, cuando no puedan surgir posibles ambigüedades.
- **R**: No-terminal que emita *hacia la derecha* (dirección 3´→ 5´), caso que se puede presentar ocasionalmente en protuberancias entre tallos o bucles en la parte derecha de la estructura (cinta 3') [13]. Se utiliza en el caso de ocurrir ambigüedades en la utilización del no terminal **L**.
- **B**: No-terminal de bifurcación que se usa para dividir varios tallos o bucles con varios ramajes que surgen de él.
- **S**: No-terminal de inicio que actúa como hijo inmediato de una derivación de bifurcación o comienzo de secuencia.
- **E**: No-terminal de finalización que termina la derivación de la secuencia.
- **D**: No-terminal de supresión, que se usa para describir una producción que no emite símbolos terminales y tampoco describe uno de los casos anteriores.

Cada uno de los no terminales tienen, por su característica estocástica, un probabilidad $e_v(a,b)$ de emitir uno o dos pares de bases. Aquí *v* es el no-terminal y $a, b \in \{A,G,C,U\}$. El símbolo $t_v(Y)$ representa la probabilidad de pasar del estado v al estado *Y*.

---

[9] Del inglés *Hidden Harkov Model*.
[10] Notación que denota la Probabilidad de que ocurra la secuencia O dado que ocurren los parámetros w.
[11] En el vocabulario molecular, se les conoce como *estructuras homólogas*.
[12] En biología molecular, 5´y 3´ son los extremos de las secuencias de ADN ARN, y tiene relevancia biológica, ya que determinan la dirección de varias funciones celulares como la duplicación y la replicación de material genético, entre otras.
[13] Teniendo como convención que una cadena se lee en la dirección 5´→ 3´.

Producción			Descripción	P (Emisión)	P(Transición)
P	→	*aYb*	De apareamiento (16 posibles pares emitidos)	$e_v(a,b)$	$t_v(Y)$
L	→	*aY*	Derivación izquierda (4 posibles símbolos emitidos)	$e_v(a,b)$	$t_v(Y)$
R	→	*Ya*	Derivación izquierda (4 posibles símbolos emitidos)	$e_v(a,b)$	$t_v(Y)$
B	→	*SS*	Bifurcación	1	1
D	→	*Y*	No derivación de símbolo	1	$t_v(Y)$
S	→	*Y*	Inicio	1	$t_v(Y)$
E	→	*ϵ*	Finalización	1	1

Figura 1. Tipos de reglas gramaticales y sus probabilidades típicas en un Modelo de Covariación[14].

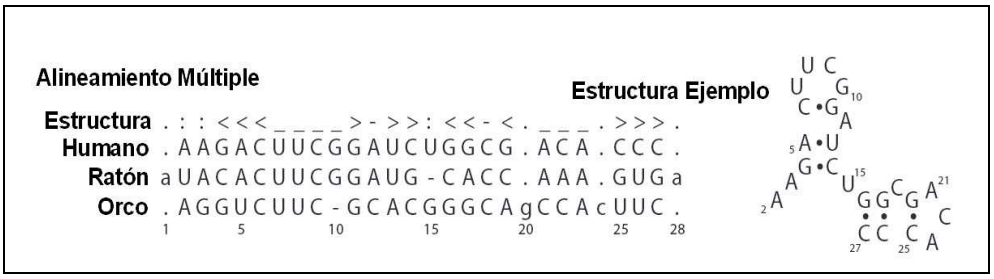


Figura 2. [15] Un alineamiento ficticio de tres secuencias donde se modelan 24 columnas consenso de 28 posibles. La secuencia de estructura anotada es consenso para la estructura de la derecha, que corresponde a la humana[16].

Para construir un CM es necesario partir de un alineamiento[17] de secuencias de ARN, con su correspondiente anotación de la estructura secundaria consenso, y las anotaciones de cuáles columnas deben ser consideradas inserciones y cuáles deben se consideradas columnas consenso (Figura 2). A partir de estas columnas se construye un árbol estructural consenso.

El CM será un grafo dirigido de M estados con transiciones dadas por $t_v(y)$, con cada estado numerado de tal manera que $(y,z) \rightarrow v$. Se puede entonces visualizar el CM como un arreglo de estados cuyas transiciones corren en una sola dirección, lo cual garantiza dos cosas: que sea posible un cálculo iterativo de programación dinámica a través de todos los estados del modelo, así como que todas las transiciones para el estado *v* pueden mantenerse como un desplazamiento y un conteo.

3.2 Implementación de la Predicción de Pseudonudos en la Estructura Secundaria como una Extensión de los Modelos de Covariación

Si se entienden los CM, y en general las SCFG, como herramientas computacionales que permiten modelar probabilísticamente relaciones distantes entre símbolos del lenguaje molecular, entonces es posible extender este concepto a los pseudonudos. Esta extensión hace posible su predicción al eliminar en la secuencia original los segmentos apareados distantes (lenguaje de copiado), y analizar normalmente la secuencia resultante.

Si la relación que existe entre los residuos de un tallo está separada por residuos no relacionados, de igual manera unos pares de residuos que forman un pseudonudo estarán separados por residuos que no necesariamente se encuentran relacionados con el pseudonudo en cuestión[18]. Por ello, así como los residuos no relacionados se separan para la determinación de la relación entre los pares de bases que forman el tallo, de igual manera es posible separar las

[14] Tomado de [6], pp 278.
[15] Este alineamiento se encuentra en formato STOCKHOLM. Para más detalles consultar: http://www.cgr.ki.se/cgb/groups/sonnhammer/Stockholm.html .
Para más formatos de secuencias en biología molecular, favor referirse a: http://www.psc.edu/general/software/packages/hmmer/manual/node38.html
[16] Tomado de [6], pp 278.
[17] Un alineamiento pretende medir cuantitativamente el grado de similitud entre dos secuencias, asignando correspondencias de elemento a elemento de una secuencia en otra(s).
[18] Se aclara que este es el punto de vista de las SCFG, alejándose considerablemente de la aproximación termodinámica, donde todas las contribuciones deberían ser tenidas en cuenta.

relaciones existentes entre los residuos no relacionados para la determinación de la relación entre los pares de bases que forman el pseudonudo.

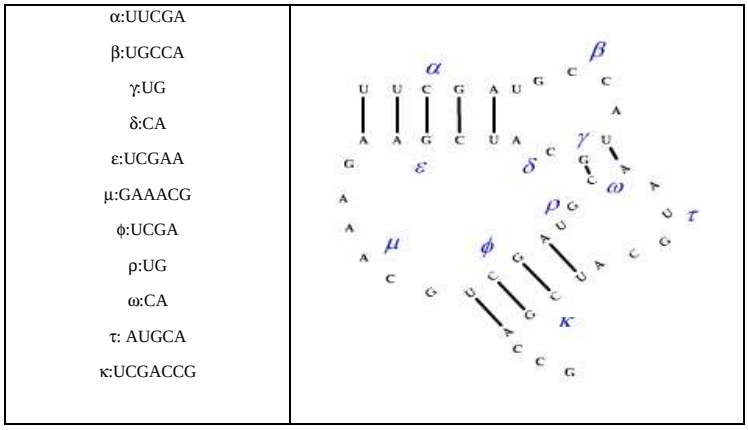


Figura 3. Visualización de una estructura pseudonudo representativa, dividida en subcadenas αβγδεμφρωτκ para posterior abstracción.

Definición. Si γ y δ son secuencias ARN tales que $|\gamma| = |\delta|$ y, siempre que $1 \leq i \leq |\gamma|$ se tiene que γ_i y $\delta^{-1}{}_i$ son bases complementarias, donde δ^{-1} es la secuencia δ escrita en sentido inverso, entonces se dice que γδ es un palíndromo biológico.

Nótese que, si x es una secuencia ARN que puede ser escrita como la concatenación de cinco subsecuencias, esto es x = πγψωσ (π y σ posiblemente vacías), de manera que γω es un palíndromo biológico, entonces el modelo CM debe predecir el apareamiento (γ,ω^{-1}) como parte de la estructura secundaria de x (ver figura 4).

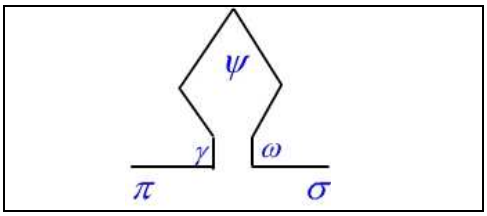


Figura 4. Visualización de la abstracción de la estructura pseudonudo dividida en subcadenas πγψωσ para análisis secuencial.

Ahora bien, si π, ψ o σ se pueden descomponer como la concatenación de cinco subcadenas αβμνρ de tal manera que βν es un palíndromo biológico, entonces el apareamiento (β,ν^{-1}) también es predicho por el modelo CM como parte de la estructura secundaria de x.

Por otra parte, si π, ψ y σ no pueden ser descompuestas como en el párrafo anterior, entonces es posible inferir que en x = πγψωσ no se encuentran apareamientos que no propongan pseudonudos en la estructura secundaria de x.

Sea x' = πψσ, la secuencia que se obtiene de x al eliminar todos los pares de bases apareados según el modelo CM. Si existen secuencias π', γ', ψ', ω' y σ' tales que x' = πψσ = π'γ'ψ'ω'σ' y γ'ω' es un palíndromo biológico, entonces el apareamiento (γ',ω'^{-1}), que hace parte de la estructura secundaria de x', denuncia un pseudonudo en la estructura secundaria de x.

Esto puede ser concluido gracias a que sólo dos casos tienen sentido:

$$(1) \qquad \gamma' \subseteq \pi \wedge \omega' \subseteq \psi$$
$$(2) \qquad \gamma' \subseteq \psi \wedge \omega' \subseteq \sigma.$$

Así las cosas, se pretende que π, ψ, σ conserven la información anterior (el análisis previo) mientras se analizan las relaciones entre γ y ω. De esta manera, el problema se subdivide y se puede hacer una segunda iteración del análisis para encontrar los dos niveles de relaciones por separado, para luego fusionarlos y dar el análisis completo al final. En

síntesis, en vez de hacer un análisis en paralelo como en [4], se trata de hacer un análisis secuencial de dos instancias del mismo problema, con la ganancia evidente en materia de recursos, los cuales antes estaban compartidos y ahora se pueden dedicar enteramente a cada una de las dos partes. Además, se reduce la escala de complejidad estructural en ambos casos de manera dramática.

3.3 Descripción General de la Metodología para la Validación de la Extensión Propuesta

Para comprobar el algoritmo propuesto se implementó un programa escrito en C/C++, que corrió en dos plataformas, divergiendo sólo en el tiempo de análisis requerido. La implementación más rápida se hizo en plataforma INTEL/Linux, procesador de 1.8GHz, 1024 Mbytes en RAM. La segunda implementación fue hecha en una plataforma IRIX 6.5, (SGI Origin 2000), con 4 procesadores en paralelo, 1Gbyte en RAM[19]. Dicho programa analiza un alineamiento en formato STOCKHOLM[20] aumentado con un marcador, *PK_cons*[21], que indica las relaciones de pseudonudos según el estándar implementado por *PseudoBase* [25]. A manera de ejemplo, la figura 5 muestra un alineamiento teórico con un pseudonudo consenso para las secuencias mostradas.

```
# STOCKHOLM 1.0
#=GF ID   trnaDummy
#=GF DE   tomado de [Eddy 2002], con PK_cons indicando pseudonudo ficticio

DF6280              GCGGAUUUAGCUCAGUUGGG.AGAGCGCCAGACUGAAGAUCUGGAGGUCC
DE6280              UCCGAUAUAGUGUAAC.GGCUAUCACAUCACGCUUUCACCGUGGAGA.CC
DD6280              UCCGUGAUAGUUUAAU.GGUCAGAAUGGGCGCUUGUCGCGUGCCAGA.UC
DC6280              GCUCGUAUGGCGCAGU.GGU.AGCGCAGCAGAUUGCAAAUCUGUUGGUCC
DA6280              GGGCACAUGGCGCAGUUGGU.AGCGCGCUUCCCUUGCAAGGAAGAGGUCA
#=GC SS_cons        <<<<<<<..<<<<.........>>>>.<<<<<.......>>>>>.....<
#=GC PK_cons        <<<[[<<..<<<<.........>>[>.<<<<<.......>>>>>.....]
#=GC RF             xxxxxxxxxxxxxxxxxxx.xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

DF6280              UGUGUUCGAUCCACAGAAUUCGCA
DE6280              GGGGUUCGACUCCCCGUAUCGGAG
DD6280              GGGGUUCAAUUCCCCGUCGCGGAG
DC6280              UUAGUUCGAUCCUGAGUGCGAGCU
DA6280              UCGGUUCGAUUCCGGUUGCGUCCA
#=GC SS_cons        <<<<.......>>>>>>>>>>>>.
#=GC PK_cons        <<<<.......>>>>>>>>]]>>.
#=GC RF             xxxxxxxxxxxxxxxxxxxxxxxx
//
```

Figura 5. Alineamiento de 5 fragmentos de ARN. Nótese que se utilizan <> para indicar relaciones de bucles, y [ ] para indicar relaciones de pseudonudos[22].

El software implementado abstrae los dos niveles de estructura, secundaria y de pseudonudos, a dos modelos separados, uno para indicar la estructura secundaria *SS_cons* (Figura 6) y otro para indicar la estructura de pseudonudos *PK_cons* (Figura 7) abstrayendo la ya analizada.

```
# STOCKHOLM 1.0
#=GF ID   trnaDummy
#=GF DE   tomado de [Eddy 2002], con PK_cons indicando pseudonudo ficticio

DF6280              GCGGAUUUAGCUCAGUUGGG.AGAGCGCCAGACUGAAGAUCUGGAGGUCCUGUGUUCGAUCCACAGAAUUCGCA
DE6280              UCCGAUAUAGUGUAAC.GGCUAUCACAUCACGCUUUCACCGUGGAGA.CCGGGGUUCGACUCCCCGUAUCGGAG
DD6280              UCCGUGAUAGUUUAAU.GGUCAGAAUGGGCGCUUGUCGCGUGCCAGA.UCGGGGUUCAAUUCCCCGUCGCGGAG
DC6280              GCUCGUAUGGCGCAGU.GGU.AGCGCAGCAGAUUGCAAAUCUGUUGGUCCUUAGUUCGAUCCUGAGUGCGAGCU
DA6280              GGGCACAUGGCGCAGUUGGU.AGCGCGCUUCCCUUGCAAGGAAGAGGUCAUCGGUUCGAUUCCGGUUGCGUCCA
#=GC SS_cons        <<<<<<<..<<<<.........>>>>.<<<<<.......>>>>>.....<<<<<.......>>>>>>>>>>>>.
#=GC RF             xxxxxxxxxxxxxxxxxxx.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figura 6. Primer Nivel de Abstracción. Nótese que los [ ] que indicaban las relaciones de pseudonudos se han abstraído con <> para completar los bucles existentes.

[19] Las diferencias en velocidad se deben principalmente a que el algoritmo es intensivo en memoria y que el segundo computador es multiusuario.
[20] Para un descripción detalla de este formato, ver : http://www.cgr.ki.se/cgb/groups/sonnhammer/Stockholm.html
[21] Este marcador no es estándar, y sólo se utiliza en este trabajo.
[22] Para mayor legibilidad, en resumen un formato Stockholm tiene un encabezado con información relevante, luego vienen el nombre y la secuencia de nucleótidos, y por último las relaciones consenso. El marcador RF indica que todas las columnas del alineamiento deben tomarse en cuenta para el modelo.

```
# STOCKHOLM 1.0
#=GF ID   trnaDummy
#=GF DE   tomado de [Eddy 2002], con PK_cons indicando pseudonudo ficticio

DF6280              GCGGAUUUAGCUCAGUUGGG.AGAGCGCCAGACUGAAGAUCUGGAGGUCC
DE6280              UCCGAUAUAGUGUAAC.GGCUAUCACAUCACGCUUUCACCGUGGAGA.CC
DD6280              UCCGUGAUAGUUUAAU.GGUCAGAAUGGGCGCUUGUCGCGUGCCAGA.UC
DC6280              GCUCGUAUGGCGCAGU.GGU.AGCGCAGCAGAUUGCAAAUCUGUUGGUCC
DA6280              GGGCACAUGGCGCAGUUGGU.AGCGCGCUUCCCUUGCAAGGAAGAGGUCA
#=GC SS_cons        ...<<...................<.......................>
#=GC RF             xxxxxxxxxxxxxxxxxxxx.xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

DF6280              UGUGUUCGAUCCACAGAAUUCGCA
DE6280              GGGGUUCGACUCCCCGUAUCGGAG
DD6280              GGGGUUCAAUUCCCCGUCGCGGAG
DC6280              UUAGUUCGAUCCUGAGUGCGAGCU
DA6280              UCGGUUCGAUUCCGGUUGCGUCCA
#=GC SS_cons        ...................>>...
#=GC RF             xxxxxxxxxxxxxxxxxxxxxxxx
//
```

Figura 7. Segundo Nivel de Abstracción. Nótese que los <> que indicaban las relaciones secundarias se han abstraído con puntos, y los [ ] que indicaban la relación de pseudonudo se han abstraído a <>, haciendo posible el análisis de estructura secundaria de los pseudonudos.

Una vez procesada la información a estos dos niveles, se procede a analizarlos por separado para poder construir los modelos consensos de ambos niveles de estructura. Con estos dos modelos construidos es posible predecir la estructura de una cadena de nucleótidos sin ninguna estructura incluida (Figura 8).
El software comparará por separado la misma cadena con respecto a los dos modelos consenso, prediciendo la estructura secundaria que probablemente tendría esta secuencia, así como el probable pseudonudo que contendría, éste sin embargo abstraído en estructura secundaria. Luego se unifican las dos informaciones, construyendo la estructura de pseudonudo para dar el resultado final con la estructura completa, tanto la secundaria como la de pseudonudo (Figura 8).

Secuencia sonda en formato FASTA[23], con su identificador seguido de la secuencia de nucleótidos.

```
>trna_yeast_phe
GCAGAUUUAGCUCAGUUGGCAGAGCGCCAGACUGAAGAUCUGGAGGUCCU
GUGUUCGAUCCACAGAAUACGCU

# STOCKHOLM 1.0
#=GF AU    Infernal 0.55

DF6280              GCGGAUUUAGCUCAGUuGGGAGAGCGCCAGACUGAAGAUCUGGAGGUCCU
#=GC SS_cons        (((((((,,<<<<___.____>>>>,<<<<<_______>>>>>,,,,,<<
#=GC PS_cons        (((([[((,,<<<<___.____>>[>,<<<<<_______>>>>>,,,,,]<
#=GC RF             gccgauaUagcgcAgU.GGuAgcgcgccacccUgucaagguggAGgUCcg

DF6280              GUGUUCGAUCCACAGAAUUCGCA
#=GC SS_cons        <<<<_______>>>>>)))))))):
#=GC PS_cons        <<<<_______>>>>>))))]])):
#=GC RF             gggUUCGAUuccccguaucggcg
//
//
```

Figura 8. Resultado de la predicción de Estructura secundaria de la secuencia trna_yeast_phe.fa. Nótese tanto la estructura secundaria <> como la estructura de pseudonudo, representada con los paréntesis "[ ]"[24].

[23] Para información detallada del formato FASTA, dirgirse a: http://www.ebi.ac.uk/help/formats_frame.html
[24] El resultado se presenta en formato WUSS modificado, también no estándar, pues este no acepta [ ] como indicador de pseudonudo. Los símbolos de puntuación indican residuos no apareados en diferentes regiones de la secuencia.

4. Resultados y Evaluación Experimental

El conjunto de secuencias fuente para la comprobación tiene el mismo origen del utilizado en [4][25]. Este conjunto ya está debidamente alineado y anotado para la estructura secundaria y de pseudonudo, y se encuentra en la *tmRNA Database*[26] [27]. El tmRNA contiene hasta 4 pseudonudos debidamente anotados. De esta base se descargaron 35 secuencias, las cuales se organizaron en un árbol filogenético, de tal manera que al crear el grupo para entrenamiento del modelo y el grupo para predicción de estructura estuvieran compensados a nivel filogenético, sin ninguna clase sobre-representada.
Este archivo, luego de ser alineado, eliminándose las columnas no compartidas entre éstas y otras secuencias de bacterias, resulta en un alineamiento completo, adecuado para ser sujeto al análisis bioinformático en cuestión.
A partir de las 5 secuencias de bacterias que pertenecen al phylum[27] de la bacteria cuya secuencia será analizada, se construyó el modelo adecuado para las dos estructuras abstraídas. A partir de estos modelos se predijo la siguiente estructura a partir de una secuencia sin anotación estructural (Figura 9).

```
Secuencia sin anotación estructural (FASTA):
>Esch.coli v-2.       741 bp
GGGGCUGauucuggauucgaCGGGAUUUgcgaaAcCCaaGGUGAUGCCGAGGgGCGGUUggCCUCGua
aaaAGCCGCaaaaaauagucgcaaacgacgaaaacuacGCUUUAGCAGcuuaauaacCUGCuUAGAGC
CCUCUCUCCCUagCCUccgCUCUUAGGacGGGGaucaAGAGAGGucaaacccaaAAGAGaucGCGUgg
aAGCCcuGCCUGGGGUUgaaGCGUuaaaacuuaaUCAGGCuaGUUUGUUAGUgGcGuGUccGUCCaCa
GCUGGCAAGCgaauguaaaGACurACuaaGCAUGUaguACCgaGGaUguagAAAUUUCGgacGCGGGu
ucaacuCCCGCCAGCUCCacca

Fragmento de la estructura predicha por el algoritmo (de la base 01 a
la 100):

Esch.coli          GGGGCUGAUUCUGGAUUCGACGGGAUUUGCGAAACCCAAGGUGAUGCCGA
#=GC SS_cons       {{{{{{{,,,,,,,,,,,,,[[[[[[[[-----[[[[[[[[[[[[[[[<<
#=GC PS_cons       <<<<<<<,,,,,,,,,,,,,<<<<<<<<-----<<<<<<<<<<<<<<<[[[
#=GC RF            ggggcuguuuucGGuUUCGAcaggauuauCgAAaccaaaagugaugccga

Esch.coli          GGGGCGGUUGGCCUCGUAAAAAGCCGCAAAAAAUAGUCGCAAACGACGAA
#=GC SS_cons       <<-<<<<<<<_>>>>------>>>>>>>,,,,,,,,,,,,,,,,,,,,,
#=GC PS_cons       [[-[[<<<[<_>>>>----]]]]]]],,,,,,,,,,,,,,,,,,,,,,
#=GC RF            ggGgcgguuggccucGuaAaaagccgcAaaAAAUAguCGCAAAcaAaaAA

(. . .)
```

Figura 9. Secuencia de bases y estructura predicha de E. coli construido a partir del alineamiento recuperado de la tmRNA DataBase[28].

Si se compara la estructura predicha con respecto a la que provee la base de datos *tmRNA Database* (figura 10), se encuentra que el algoritmo predice en general toda la estructura, correctamente ubicando los dos pseudonudos que se encuentran, si bien presenta algunas inconsistencias en el tamaño del pseudonudo o las bases que lo conforman en alguna región de toda la molécula. En ningún caso se predijeron falsos positivos, es decir, un pseudonudo donde no debería estar. Esto comprueba que el algoritmo y su prototipo son capaces de analizar grandes cantidades de datos de una manera relativamente confiable, superando a otros algoritmos que analizaban pseudonudos, que si bien también tenían similares resultados, sólo se podían analizar fragmentos mucho más pequeños, un solo pseudonudo, o ignorando por completo la estructura secundaria que no se relacionaba con el pseudonudo [4].

[25] Tanto el prototipo del software implementado (bajo licencia GNU) como las fuentes biológicas utilizadas están a disposición de quién las solicite en m.nino68@egresados.uniandes.edu.co.
[26] Acceso online en: http://psyche.uthct.edu/dbs/tmRDB/tmRDB.html
[27] División taxonómica utilizada en biología para clasificar los organismos.
[28] http://psyche.uthct.edu/dbs/SRPDB/SRPDB.html.

```
Fragmento de la estructura real por el algoritmo (de la base 01 a la 100):

Esch.coli        GGGGCUGauucuggauucgaCGGGAUUUgcgaaAcCCaaGGUGCAUGCCGAGGgGCGGUUggCCUCGuaaaa
#=GC SS_cons     <<<<<<<-------------<<<<<<<-----<<<<<<<<<<<<<<<<<<<<-<<<<<<<-<<->>----->
#=GC PS_cons     ------------------------------------------------((((((((((((udoknot-I))

Esch.coli        AGCCGCaaaaaauagucgcaaacgacgaaaacua
#=GC SS_cons     >>>>>>----------------------------
#=GC SS_cons     ))))))----------------------------
(...)
//
```

Figura 10. Estructura real de E. coli recuperada en la tmRNA DataBase[29].

Es importante anotar que los sistemas conocidos a la fecha tienen éxito en la identificación de pseudonudos simultáneamente con secuencias cercanas a las 100 pares de bases [4][5]. La secuencia anterior tiene más de 740 pares de bases, lo que habla por si mismo del aumento significativo en el alcance práctico que este algoritmo presenta para el análisis de pseudonudos.

5. Conclusiones

Dada la ingente cantidad de datos que producen los proyectos genómicos, se hace preponderante la necesidad de encontrar nuevos filtros sensibles para encontrar las secuencias adecuadas. Dado que en algunos casos hace falta más información para analizar las relaciones entre secuencias, estos filtros necesitan de nuevas características con las cuales comparar secuencias. Uno de los factores que apenas se comienza explotar para tal fin son los pseudonudos, por lo que el algoritmo presentado aquí también es una contribución en ese sentido.

Para el caso particular del ARN y su predicción de estructura secundaria, aquí se intenta desacoplar y desagregar un problema complejo, dividiéndolo en subproblemas de más fácil análisis, con el cuidado y atención que requiere conservar la coherencia y de manera rigurosa volver a cohesionar la información. Dado que los resultados fueron satisfactorios, se confirma el éxito de esta aproximación integrada, aproximaciones que ya han tenido gran éxito en otros campos de la bioinformática[30].

Sin embargo, independientemente de los resultados obtenidos, el algoritmo planteado en este trabajo sugiere una aproximación coherente para ser aplicada a otros problemas bioinformáticos que adolecen de la misma complejidad. Por esta razón se propone, como trabajo futuro, aplicar esta heurística en dichos problemas que abundan en bioinformática. Esta reorganización de los problemas abstrayéndolos de tal manera que se aprovechen al máximo los recursos y se disminuya la complejidad inherente, será definitivamente una heurística que agregará gran valor al análisis que se quiera realizar.

Agradecimientos

A Carlos Jaramillo, por sus múltiples y pacientes conversaciones sobre biología molecular, y a Rafael Gómez, por sus inspiradoras sugerencias sobre una visión integrada de los sistemas y el conocimiento en general.

Referencias

[1] Baldi, P.; Brunak, S., *Bioinformatics: The machine learning approach.* 2nd edition. Ed. *MIT Press*. (2001).

[2] Brown, M.; Wilson, C., RNA Pseudoknot Modeling Using Intersections Of Stochastic Context Free Grammars with Applications to Database Search. In Hunter, L. and Klein, T., editors. *Pacific Symposium on Biocomputing*, (1995), pp. 109-125.

[3] Brown, M., RNA Modeling Using Stochastic Context-Free Grammars. Phd´s Thesis. University of California, Santa Cruz. (1999).

[4] Cai, L.; Russell, L.; Wu, Y., Stochastic modelling of RNA pseudoknotted structures: a grammatical approach. *Bioinformatics* vol 19 (2003), pp. i66-i73.

[5] Deogun, J.S.; Donis, R.; Komina, O.; Ma, F., RNA Secondary Structure Prediction with Simple Pseudoknots. In Proc. Second Asia-Pacific Bioinformatics Conference (APBC2004), Dunedin, New Zealand. *CRPIT*, 29.

[29] http://psyche.uthct.edu/dbs/SRPDB/SRPDB.html.
[30] Por ejemplo, en la predicción de estructura proteica [6].

Chen, Y.-P. P., Ed. ACS. (2004), pp. 239-246.

[6] Durbin, R.; Eddy, S.; Krogh, A.; Mitchinson, G., Biological sequence analysis: Probabilistic models of proteins and nucleic acids. *Cambridge University Press*. (1998).

[7] Eddy, S. R., A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics*. (2002) Vol 3 pp. 18 http://www.biomedcentral.com/1471-2105/3/18.

[8] Felden, B.; Massire, C.; Westhof, E.; Atkins, J.; Gesteland, R., Phylogenetic analysis of tmRNA genes within a bacterial subgroup reveals a specific structural signature. *Nucleic Acids Res.*, vol 29 (2001), pp. 1602-1607.

[9] Gultyaev, A.P.; van Batenburg. F.; Pleij, C., The Computer simulation of RNA folding pathways using a genetic algorithm. *J. Mol. Biol.*, 250 (1995), pp. 37-51.

[10] Hopcroft, J.E.; Ullman, J.D., *Introduction to automata theory, lenguage and computation*. Addison-Wesley. (1979).

[11] Lari, K.; Young, S., The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 4 (1990), pp. 35-56.

[12] Lee, D.; Han, K., Prediction of RNA Pseudoknots – Comparative Study of Genetic Algorithms. *Genome Informatics* 13 (2002), pp. 414-415.

[13] Lewis, H.R.; Papadimitriou, Ch.H., *Elements of the theory of computation*. Prentice-Hall. (1981).

[14] Lyngsø, R.B.; Pedersen, C.N.S., (2000) Pseudoknot Prediction in Energy Based Models. To appear in *Journal of Computational Biology*.

[15] Paillart, J.; Skripkin, E.; Ehresmann, B.; Ehresmann, C.; Marquet, R., In vitro evidence for a long range pseudoknot in the 5'-untranslatd and matrix coding regions of HIV-1 genomic RNA. *J. Biol. Chem*., Vol 277 (2002), pp. 5995-6004.

[16] Pleij, C.W.A.; Rietveld, K.; Bosch, L., A new principle of RNA folding based on pseudoknotting. *Nucleic Acids Res*. Vol 13,5 (1985), pp. 1717-1731.

[17] Rivas E.; Eddy S., A Dynamic Programming Algorithm for RNA Structure Prediction Including Pseudoknots. *Journal of Molecular Biology*, Vol. 285, no. 5 (1999), pp. 2053-2068(16).

[18] Rivas, E.; Eddy, S., The language of RNA: a formal grammar that includes pseudoknots. *Bioinformatics* 16 (2000), pp. 334-340.

[19] Ruan, J.; Stormo, G.; Zhang, W., An Iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics*. Vol 20 (2004), pp.58-66.

[20] Searls, D., The Computational linguistics of Biological sequences. In: *Artificial Intelligence and Molecular Biology*, chap 2. *AAAI Press*. (1992).

[21] Shannon, C.E., A mathematical theory of communication. *Bell system Technical Journal*, Vol 27 (1948), pp. 379-423, 623-656.

[22] Strickberger, M., *Evolution*. 3rd Edition. Jones and Bartlett Publishers (2000).

[23] Tabaska, J.; Cary, R.; Gabow, H.; Stormo, G., An RNA folding method capable of identifying pseudoknots and base triples. *Bioinformatics Journal*,Vol 14 (1998), pp. 691-699.

[24] Tahi, F.; Engelen, S.; Régnier, M., Pcdfold-An algorithm for prediction of RNA including all kinds of Pseudoknots. Computers and Chemsitry, Vol 26 (2002), pp. 521-530.

[25] Van Batenburg, F.; Gultyaev, A.; Pleij, W.; Ng, J.; Oliehoek, J., PseudoBase, a database with RNA pseudoknots. *Nuc. Ac. Res.*, Vol 28 (2000), pp. 201-204.

[26] Zuker, M., On finding al suboptimal foldings of an RNA molecule. *Science*, 244 (1989), pp.48-52.

[27] Zwieb, C.; Gorodkin, J.; Knudsen, B.; Burks, J.; Wower, J., tmRDB (tmRNA database). *Nucleic Acids Research*, vol 31, No.1, (2003), pp. 446-447.

Geração Automática de Interfaces para Aplicações Extensíveis por Usuários Finais

Sérgio Roberto P. da Silva e Marco Aurélio Jeronymo
Universidade Estadual de Maringá — Departamento de Informática
Maringá - PR, Brasil, 87020-900
{srsilva, marcoaj}@din.uem.br

Abstract

Current extensible applications do not take into account the user's difficulties to create the graphic interfaces which are necessary to their extensions. In our work, we propose a system for the automatic generation of interfaces in extensible applications. The interfaces which are required for an extension are expressed in an extension language that allows the specification of interaction elements. The proposed system has a set of intelligent software agents that uses this code and an application design knowledge base for the generation of extension's interfaces. This knowledge base has a set of interface design rules (guidelines and styleguides) and rules defined by the application's design that will be used for choosing the elements to compose the user's defined interface. We adopt the Semiotic Model for extensible application as a base for the system architecture.

Keywords: *Automatic Interface Generation*, *End-User Extensible Applications*, *Semiotic Model*.

Resumo

As aplicações extensíveis atuais não levam em consideração as dificuldades encontradas pelos usuários na criação das interfaces gráficas necessárias às suas extensões Neste trabalho, propomos um sistema para a geração automática de interfaces em aplicações extensíveis. As interfaces necessárias à extensão são expressas em uma linguagem de extensão que permite a especificação de elementos de interação. O sistema proposto tem um conjunto de agentes inteligentes de software que utilizam este código e uma base de conhecimento do design da aplicação para a geração das interfaces das extensões. Esta base de conhecimento contém um conjunto de regras de design de interfaces (*guidelines* e *stylesguides*) e de regras definidas pelo designer da aplicação que serão empregadas na escolha dos elementos que comporão a interface definida pelo usuário. Como base para a arquitetura do sistema proposto, foi empregado o Modelo Semiótico para aplicações extensíveis.

Palavras Chaves: Geração automática de interfaces, Aplicações extensíveis por usuários finais, Modelo Semiótico.

1. INTRODUÇÃO

Observa-se, há algum tempo, que a indústria de software vem aumentando a quantidade de funcionalidades nas aplicações, tentando, dessa forma, satisfazer as necessidades da maioria dos usuários. Todavia, isto tem implicado em softwares cada vez maiores, mais complexos e, conseqüentemente, com maiores problemas de usabilidade. Diversas vezes tais funcionalidades sequer são utilizadas, pois é necessário muito conhecimento e um grande dispêndio de tempo no aprendizado por parte do usuário. Uma alternativa, objetivando resolver esse problema, é permitir que os próprios usuários configurem e alterem o software para que este venha satisfazer suas necessidades específicas.

Na literatura, podemos encontrar várias designações para a capacidade de alteração de um software, dentre as quais podemos citar: customização, parametrização, adaptação, programação por usuário final, extensão, personalização e *tailoring* [4]. Neste artigo, utilizaremos a palavra **extensão** como termo genérico para designar esta capacidade. As extensões podem ser realizadas basicamente de duas formas: através do uso da linguagem de interface da aplicação e através do uso de uma linguagem de extensão que trabalhe embutida no software [4]. Como exemplo da primeira, podemos citar as pesquisas em programação por demonstração [1] [18] [13], e da segunda, podemos citar as linguagens de macro embutidas em softwares como, por exemplo, *Hypercard*™ [9] e *Visual Basic*™ [17].

Apesar de a segunda abordagem citada ser a mais comum hoje em dia, esta solução em geral esbarra no problema de falta de conhecimento de programação por parte do usuário. Deste modo, é essencial que o usuário seja apoiado na tarefa de criação dessas extensões por meio da disponibilização de uma linguagem que seja de fácil aprendizado e uso e de um ambiente de extensão que o guie no processo de criação. Entretanto, na maioria dos ambientes de extensão

atualmente em uso este apoio é fraco e não leva em conta as dificuldades do usuário para criar as interfaces gráficas necessárias às suas extensões [9] [17]. Geralmente, o usuário tem que escrever linhas de códigos para descrever as interfaces em vez de construí-las graficamente, processo ao qual ele está habituado. Mesmo quando é possível descrever a interface graficamente, o usuário tem que fazer a ligação da interface com o código e isto novamente implica na necessidade conhecimentos de programação, o qual, ele comumente não possui.

Da Silva [4] propõe um modelo para o problema da criação de extensões ao software pelos usuários finais, o Modelo Semiótico para aplicações extensíveis. Neste trabalho, **usuário final** é todo usuário que consiga utilizar uma aplicação para a realização de uma tarefa. A arquitetura de software do Modelo Semiótico contempla um sistema de extensão composto por um conjunto de agentes de software responsáveis pela interpretação da linguagem de extensão e pela manutenção dos princípios teóricos que sustentam o modelo e um conjunto de agentes de software construtores de interface.

Visto que descrever elementos de interface na forma textual no código na extensão é uma tarefa muito difícil para os usuários finais, que estão habituados a tão somente utilizar estes elementos. Deste modo, nosso trabalho propõe o desenvolvimento de um conjunto de agentes de software que auxiliem o usuário final na geração automática de interfaces para aplicações extensíveis, facilitando, assim, a tarefa de criação destas interfaces.

É importante ressaltar que este trabalho trata somente de sistemas adaptáveis, ou seja, sistemas que permitem ao usuário configurar suas preferências, interagindo direta e explicitamente com a aplicação. Portanto, não abordamos os sistemas adaptativos, os quais alteram o próprio conteúdo ou estado autonomamente em resposta às interações do usuário, criando, desta forma, extensões à aplicação automaticamente [7].

Este trabalho faz parte de um projeto denominado APEX [3], que tem como objetivo a construção de aplicações extensíveis que sigam a proposta do Modelo Semiótico. Assim, na seção 2 apresentamos a base teórica utilizada neste trabalho. Em seguida, na seção 3, avaliamos os trabalhos relacionados e a ontologia de elementos de interface empregada neste trabalho. Depois, na seção 4, explicamos os conjuntos de conhecimento que são utilizados para a seleção de elementos de interface e discutimos o funcionamento do sistema proposto. Finalmente, na seção 5, discutimos a contribuição de nosso trabalho e propomos alguns caminhos que poderão ser seguidos no futuro.

2. O MODELO SEMIÓTICO DE APLICAÇÕES EXTENSÍVEIS

As aplicações extensíveis permitem aos usuários finais alterarem suas funcionalidades para que a aplicação satisfaça suas reais necessidades de uso. O Modelo Semiótico baseia-se na Engenharia Semiótica [6] e no Modelo de Comunicação Verbal de Jakobson [10] para fundamentar a definição de um modelo teórico; um processo para a realização de extensões ao software; uma linguagem de extensão para programação para usuários finais; e uma arquitetura de software. Na Engenharia Semiótica uma aplicação extensível é vista como uma mensagem única e unidirecional na qual o designer do software está comunicando ao usuário as seguintes informações:

- Escopo de problemas que o software está apto a resolver, ou seja, qual o modelo de funcionalidade da aplicação;
- Como é possível interagir com a aplicação para resolver estes problemas, ou seja, qual é o modelo de interação da aplicação; e
- Quais as possibilidades de extensão desta solução, ou seja, qual o modelo de extensibilidade da aplicação.

O Modelo Semiótico possui dois princípios teóricos que devem ser seguidos para que as extensões geradas mantenham a consistência com o resto da aplicação, são eles: o princípio da Abstração Interpretativa e o princípio do Contínuo Semiótico [5]. O primeiro princípio procura avaliar quão bem um código artificial abstrai outro sobre o qual ele está implementado. Assim, podemos dizer que, quando o usuário interage com um software, é necessário que ele consiga interpretar os códigos e elementos empregados na interface, recorrendo apenas ao conhecimento disponível no contexto ao qual está interagindo. Caso ele não consiga interpretar algum elemento empregado e necessite de conhecimentos em nível de implementação do software, podemos dizer que a Abstração Interpretativa foi quebrada. O segundo princípio procura avaliar o quanto dois códigos artificiais apresentam barreiras à sua tradução mútua, ou seja, procura avaliar a tradução de um código artificial para outro, observando se os textos traduzidos são válidos e pragmaticamente adequados no código destino. Portanto, ele avalia as barreiras existentes para traduzir extensões criadas na linguagem de programação por usuários finais (LPUF) em construções funcionais e usáveis da linguagem de interface do usuário (LIU). Veremos a seguir mais detalhadamente os elementos que compõem o Modelo Semiótico.

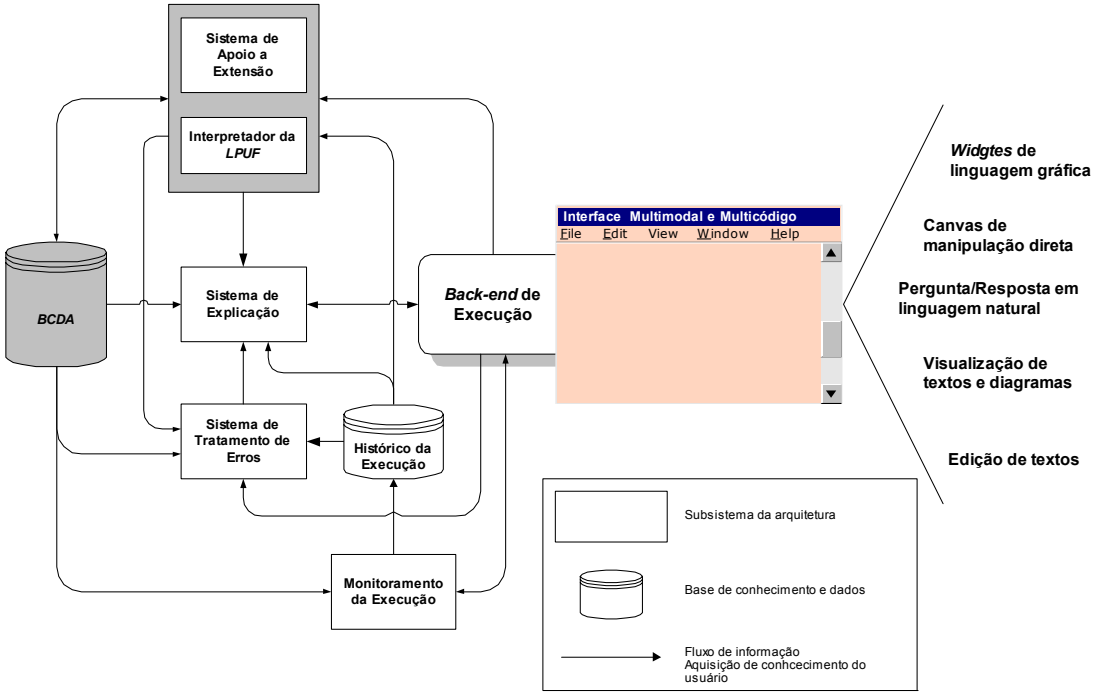


Figura 1. Arquitetura do Modelo Semiótico.

2.1. Arquitetura do Modelo Semiótico

A arquitetura de software do Modelo Semiótico pode ser visualizada na Figura 1. Ela está dividida em duas partes: na primeira, temos o *back-end* de execução e a interface multimodal e multicódigo, que são os elementos comuns às aplicações atuais; na segunda, temos os elementos restantes que diferenciam uma arquitetura para aplicações extensíveis que segue o Modelo Semiótico das que não o seguem. Esses elementos são: um sistema de tratamento de erros, que faz uso do histórico de execução, criado por um agente de software de monitoramento da execução, em conjunto com uma descrição da tarefa atual para identificar o contexto de ocorrência do erro; um sistema de explicação, que auxilia o usuário no esclarecimento de dúvidas sobre o modelo de usabilidade e extensibilidade do software; uma base de conhecimento do design da aplicação (BCDA), que é empregada na resolução das referencias da LPUF e na manutenção dos princípios teóricos da abstração Interpretativa e do Contínuo Semiótico que sustentam o modelo; e um sistema de extensão, que permite a criação de extensões pelos usuários finais. Todos esses elementos são essenciais para capacitar o usuário final na criação de extensões.

Assim, quando o usuário tem a intenção de gerar uma extensão, ele utilizará o sistema de apoio à extensão para escrever o código da extensão na LPUF e, no mínimo, gerar um elemento de interação para a sua ativação pela interface. O sistema de extensão é auxiliado por um agente de software que realiza a interpretação de textos do código da extensão. Apesar de importante este agente não é o foco deste trabalho e mais detalhes poderão ser encontrados em Da Silva [2]. Além deste agente, o sistema de extensão conta com um grupo de agentes de software construtores de interface que auxiliam o usuário na geração de elementos de interface ao criar sua extensão.

Para realizar a geração automática de interfaces, é preciso realizar a escolha de determinados elementos de interface, essa escolha é efetuada de acordo com informações adquiridas do usuário e da BCDA que, segundo o Modelo Semiótico, deve estar inclusa em todo software extensível. A BCDA é utilizada para o armazenamento do modelo do domínio da aplicação e do modelo do software. O primeiro representa as relações estáticas dos objetos do domínio da aplicação e suas interações, e o segundo representa as interações entre alguns objetos do domínio e os objetos necessários à implementação das funcionalidades do software e de sua interface, modelando a solução desenvolvida pelo designer para os problemas do usuário. Ela também é utilizada para armazenar as regras de regras de design de interfaces gráficas e também as regras de design empregadas pelo designer do software, durante a construção da aplicação, que servirão para guiar a seleção dos elementos de interface a serem gerados. Tais regras determinam a maneira pela qual os elementos de interface são organizados para criar a linguagem única de interação da aplicação. É importante salientar que, caso as regras de design não sejam seguidas, poderão ser construídas interfaces que trabalhem

de maneira totalmente incoerente com o restante da aplicação, podendo, desta forma, causar a degradação da usabilidade do software como um todo.

2.2. A Linguagem-Tipo para Programação por Usuários Finais

O Modelo Semiótico também define uma linguagem-tipo para programação por usuários finais (a LPUF) que trabalha embutida no seu ambiente de extensão. Esta linguagem-tipo permite ao usuário escrever o código das extensões e, conforme Da Silva [4], está dividida basicamente em duas partes: uma metalinguagem e um núcleo operativo.

A metalinguagem contém mecanismos para a declaração de novas entidades ou extensão de antigas e permite a criação de extensões à ontologia de um domínio específico, por meio da definição de conhecimento declarativo. Esta linguagem-tipo permite aos seus usuários a manipulação dos seguintes elementos [4]:

- Entidades — representando os elementos do domínio pertencentes ao modelo da aplicação que são visíveis e manipuláveis pelos usuários finais através da interface do software;
- Atributos — representando as características das entidades do domínio que são visíveis e manipuláveis pelos usuários através da interface do software;
- Partes — representando as entidades que compõem (fazem parte ou estão agregadas a) uma outra entidade do domínio e que são visíveis e manipuláveis pelos usuários através da interface do software;
- Ações — representando as operações que uma entidade pode realizar no domínio e que tenham seus resultados visíveis para os usuários, ou seja, somente são consideradas as operações que provocam uma modificação direta na interface do software.
- Interfaces — representando os mecanismos de ativação e os canais de comunicação que os usuários podem utilizar para ativar uma ação ou para comunicar a necessidade de dados e/ou os resultados da realização de ações; e
- Relações — representando as inter-relações entre as diferentes entidades do domínio que são visíveis e manipuláveis pelos usuários através da interface do software.

O núcleo operativo é composto de mecanismos de referenciação e de controle. Os mecanismos de referenciação possibilitam a descrição de referências a objetos, empregando uma gramática reduzida da linguagem natural que permite ao usuário o emprego de determinantes, que operam como quantificadores sobre os objetos de uma ação; ordinais, que operam como seletores para o caminhamento em objetos estruturados; de anáforas, na forma de pronomes ou determinantes, que operam como ponteiros para objetos previamente referenciados do texto; de adjetivos e substantivos pré-modificadores, e de sintagmas preposicionais, que operam como qualificadores na especificação de um atributo ou parte de um objeto. Esses mecanismos visam aumentar a coesão textual e facilitar a interpretação por parte do usuário final. Os mecanismos de controle contêm regras que possibilitam a definição de ações condicionais, de iteradores explícitos e a aplicação de ações sobre referências.

Além dos mecanismos que possibilitam a descrição de ações e entidades, a linguagem-tipo do Modelo Semiótico contém mecanismos que possibilitam a definição de elementos de interação com o usuário. Estes mecanismos são fundamentais para nosso trabalho, pois são responsáveis pela ativação dos agentes construtores de interface. Para que possamos ter um melhor entendimento da forma de ativação dos agentes de software, vamos utilizar o exemplo apresentado na Figura 2. Este exemplo reflete o código completo para a criação de uma extensão para "Envio de uma mensagem para um grupo". Ele possui apenas elementos das sublinguagens de referenciação a objetos e de controle da linguagem-tipo discutida.

```
1  To send a message to a group is an action of the e-mailer, which is activated by the menu
   and follows these instructions:

2  When the group is not indicated, ask-for it using the get-data dialog.

3  If the group belongs to the address-book, send the message to all contacts of the group.

4  If not, show the message "The specified group does not exist. Please, check your data and
   try again.".

5  When there is an error, show the message "An error has occurred during the sending of the
   messages. Please, check your data and try again." using the error-handling dialog.
```

Figura 2. Exemplo de uma extensão em código LPUF.

Podemos verificar, na primeira sentença do código do exemplo, a presença de um dos mecanismos de definição de elementos de interação com o usuário. Nesta linha está indicado (em sublinhado) que a ação criada poderá ser ativada a partir do menu da aplicação. Um outro exemplo da definição de elementos de interação está na definição de um diálogo

com o usuário — que será usado na aquisição de dados, caso esta ação seja ativada pela interface da aplicação — que ocorre na segunda linha do código (em sublinhado simples e duplo). Todas as definições de elementos de interação funcionam como *hyperlinks* para a ativação dos agentes construtores de interfaces propostos neste trabalho. Veremos maiores detalhes do funcionamento deste processo na seção 4.

Devemos salientar que a linguagem-tipo do Modelo Semiótico está descrita na língua inglesa e, visando uma integração com ela, nosso trabalho segue a mesma língua. A escolha pela língua inglesa, segundo Da Silva [4], reflete a intenção de que o trabalho possa ser avaliado por uma comunidade de informática mais ampla, sendo possível a adaptação a outras linguagens naturais.

2.3. Os Processos de Construção de Extensões

Como último elemento do Modelo Semiótico, temos a definição de um conjunto de processos para a construção de extensões. Esses processos visam a uma orientação passo a passo do usuário na realização da tarefa de extensão. O processo global de construção de extensões é dividido em etapas que são realizadas seqüencialmente e, assim, garantem a manutenção dos princípios da Abstração Interpretativa e do Contínuo Semiótico [5]. Basicamente, o processo tem o seguinte funcionamento: primeiramente, o usuário pode selecionar uma tarefa entre criar uma nova extensão e modificar ou revogar uma extensão existente. Caso a escolha seja entre criar ou modificar uma extensão, verifica-se se os elementos do texto da descrição da extensão respeitam as regras sintáticas da LPUF. Logo após, verifica-se se a descrição da extensão realizada é válida. Estas verificações são necessárias para que o processo possa garantir, até onde for possível, que toda extensão a ser inserida na BCDA esteja correta e válida. Depois, o processo realiza a documentação da extensão, responsável por manter um mínimo de explicação a ser disponibilizado na linguagem de explicação do usuário do software para a nova funcionalidade. Tal explicação é livremente expressa pelo usuário e, desta forma, não é possível garantir que ela esteja de acordo com a nova funcionalidade do software. Por último, ele atualiza a BCDA de forma automática e garante a evolução consistente da BCDA durante o processo de realização da extensão.

Além do processo global, o Modelo Semiótico define outros processos responsáveis pela construção de extensões, entre eles destacamos o processo de criação de ações nas extensões, visualizado na Figura 3. Esse processo também é

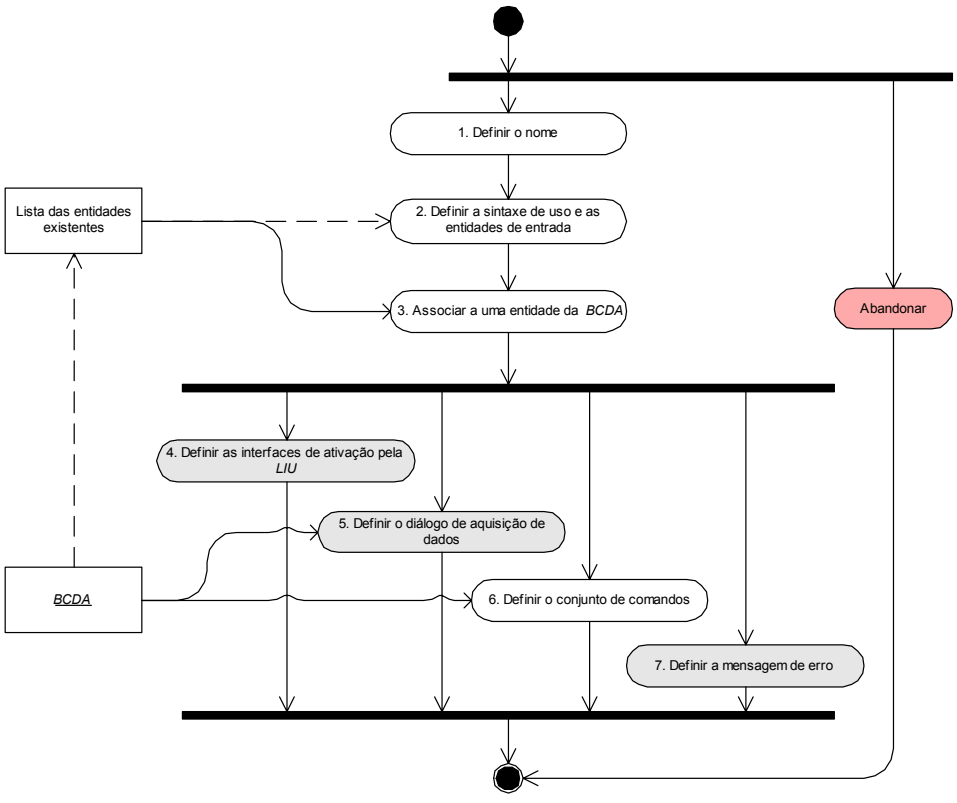


Figura 3. Processo de criação de novas ações em uma aplicação extensível.

constituído de tarefas, sendo responsável por manter a estrutura da BCDA e garantir a criação de extensões monotônicas, ou seja, não permitir a criação de elementos isolados dentro do modelo da aplicação.

As tarefas do processo apresentado na Figura 3 que mais nos interessam são as tarefas 4, 5 e 7, que são as responsáveis por auxiliar o usuário na definição de elementos de interface com a aplicação. O Modelo Semiótico define um conjunto de agentes de software para cada uma destas tarefas, os quais serão discutidos em detalhes na seção 4.

3. SISTEMAS GERADORES DE INTERFACE

Ao pesquisarmos na literatura e na Internet à procura de trabalhos correlatos, pudemos observar que são várias as designações empregadas para as ferramentas de geração de interfaces, tais como "*Toolkits*", "*User Interface Development*", "*Environments/Interface Builders*" e "*Interface Development Tools*". Myers [16] define mais especificamente esses termos e utiliza o termo geral "*User Interface Tool*" para todos os softwares que têm como objetivo criar interfaces gráficas. Baseado na maneira do designer especificar como deve ser a interface, Myers definiu quatro categorias para essas ferramentas: "*Language Based Tools*", "*Application Frameworks*", "*Model-Based Generation*" e "*Interactive Graphical Specification*".

Em especial, comentaremos a categoria "*Model-Based Generation*" que, assim como em nosso trabalho, utiliza uma forma de geração automática de interfaces. As ferramentas nesta categoria utilizam como base para a geração das interfaces modelos de interfaces predefinidos. Muitas dessas ferramentas, tais como Mickey [19], Jade [25] e DON [12] se concentram na criação de menus e caixas de diálogos. A ferramenta Jade permite ao desenvolvedor usar um editor gráfico para alteração da interface gerada, se ela não estiver de acordo com suas preferências. Uma outra ferramenta que merece ser destacada é a HUMANOID [22] que possibilita a modelagem da apresentação, do conteúdo e dos diálogos de uma interface. No entanto, é importante salientar que estas ferramentas têm como objetivo auxiliar um desenvolvedor de software a construir interfaces gráficas e que nosso trabalho está preocupado com a construção de interfaces gráficas por usuários finais. Esta diferença é fundamental, pois o nível de conhecimento destes fatores é muito distinto, vindo este fato a requerer uma abordagem bastante distinta das ferramentas acima citadas.

Como alternativa às ferramentas *model-based* encontramos o projeto Seguia, desenvolvido por Vanderdonckt [23] [24], que segue uma idéia semelhante à nossa proposta. Este projeto consiste em um sistema especialista que fornece aos desenvolvedores de sistemas interativos auxílio na produção de interfaces de usuários automaticamente. Este sistema utiliza o conceito de objetos de interação abstratos, os quais são posteriormente transformados em objetos de interação concretos ou *widgets*. O Seguia foi criado para o desenvolvimento de aplicações comuns e, por isso, utiliza regras com as diretrizes de design da interface (*guidelines*) e regras com os guias de estilo específicos (*styleguides*) das plataformas utilizadas. Com estas regras, ele cria unidades de apresentação que são consideradas como um ambiente completo e necessário para o usuário executar uma subtarefa específica de uma tarefa interativa. Deste modo, ele auxilia os desenvolvedores a construir automaticamente uma unidade de apresentação completa de interfaces de usuário através de duas estratégias de leiaute. A primeira estratégia é completamente automática, enquanto a outra é auxiliada por um assistente que interage com o desenvolvedor solicitando a aprovação de determinada escolha realizada.

Existem três pontos relevantes de diferença do Seguia com nosso trabalho. Primeiro, o personagem foco de nosso trabalho são os usuários finais. Isto implica que não consideramos necessário qualquer conhecimento de programação ou do funcionamento interno de um sistema computacional para que se possa vir a utilizar um sistema que siga os princípios propostos neste trabalho. Fato este que traz grandes conseqüências sobre a inteligência que nossos agentes terão de possuir. Segundo, utilizamos também um subconjunto de regras de design definidas pelo designer da aplicação, além do conjunto de regras empregado no Seguia. Este novo conjunto determina a maneira pela quais os elementos de interface são organizados, ou seja, estas regras impõem uma limitação ao usuário na criação de interfaces, para que o usuário não venha a causar a degradação ou destruição da mensagem original da aplicação. Por último, nosso trabalho utiliza o Modelo Semiótico de aplicações extensíveis como base do processo de criação dos elementos de interface.

3.1. Uma Ontologia de Elementos de Interface

Analisando o trabalho de Vanderdonckt [24], verificamos que o mesmo emprega uma ontologia de elementos de interface que é utilizada para a criação de interfaces. Esta ontologia também é prevista no Modelo Semiótico como parte do modelo do software presente na BCDA. No entanto, a ontologia do Seguia foi criada para ser usada por programadores e é baseada nas funções que os elementos realizam na interface, utilizando termos técnicos para a designação dos elementos. Como nosso público-alvo são usuários leigos, criando extensões limitadas ao software, verificamos que esta ontologia apresentava algumas dificuldades a estes usuários, devido à quantidade e à complexidade dos elementos de interface nela descritos. Deste modo, criamos uma ontologia centrada em usuários finais para o nosso projeto. Para a criação dessa ontologia, desenvolvemos um questionário para entrevista com questões referentes às designações utilizadas pelos usuários finais aos elementos empregados nas interfaces. Durante o processo da entrevista,

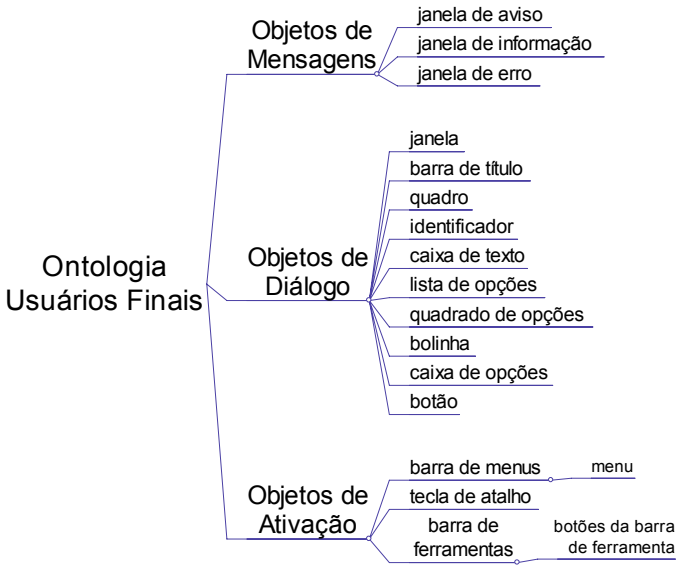


Figura 4. Ontologia de elementos de interface centrada a usuários finais.

o entrevistador indicava os elementos na interface e solicitava ao usuário a designação que ele utilizava para aquele elemento. Nesse processo, tomou-se o cuidado de não utilizar termos técnicos na formulação das perguntas para evitar direcionamento na resposta do usuário. A partir das respostas encontradas, extraímos as designações mais utilizadas pelos usuários para certos elementos da interface. Com essas designações, criamos a ontologia dos elementos de interface utilizada em nosso projeto, a qual pode ser visualizada na Figura 4.

Apesar de a pesquisa abranger um grande número de elementos de interface, optamos por escolher somente os elementos que serão disponibilizados aos usuários finais na construção de interfaces ás suas extensões. Pelo fato de somente disponibilizarmos a criação de tipos pré-definidos de interface, dividimos esta ontologia em três categorias principais de elementos, que são: os objetos de ativação (menus ou atalhos), os objetos de diálogo e os objetos de mensagens. Na categoria de objetos de ativação selecionamos os elementos que podem ser usados para ativar uma extensão por meio da interface. Na categoria dos objetos de diálogo, selecionamos elementos que são utilizados na criação de diálogos de aquisição de dados. Na categoria de objetos de mensagens, selecionamos os elementos necessários para a criação de mensagens a serem exibidas durante a execução da extensão.

4. A ARQUITETURA DO SISTEMA PROPOSTO

Com a ontologia de elementos de interface criada, podemos discutir a modelagem dos conjuntos de conhecimento utilizados neste trabalho. Esses conjuntos são compostos dos seguintes tipos de conhecimentos: as diretrizes (*guidelines*) e os guias de estilos das plataformas utilizadas (*styleguides*); as regras de design criadas pelo próprio designer do software, que são responsáveis por garantir a não degradação das interfaces; e por último, a ontologia do domínio e dos elementos de interface. Todos estes conjuntos de conhecimento estão armazenados na BCDA.

4.1. A Criação dos Conjuntos de Conhecimento

Para a criação das regras de design o designer conta com um ambiente onde pode definir a seqüência de procura das regras priorizando-as conforme sua vontade. Assim ele definirá percentuais para cada regra e, com isso, estará resolvendo possíveis problemas de conflitos de regras, quando tivermos mais que uma regra escolhida na geração de uma interface.

O primeiro subconjunto de conhecimento é baseado em diretrizes de interface amplamente utilizadas por designers ou programadores de interface e em normas de estilo. Como exemplos de diretrizes, podemos citar: as informações pertinentes à quantidade de valores que cada elemento pode exibir na interface a quantidade de valores possíveis de se escolher para determinado elemento, etc. Tais diretrizes são disponibilizadas por grandes empresas que adotam padrões de interface amplamente utilizados pelos usuários, tais como *Microsoft Corporation*® [14] e *Sun Microsystems*® [21], constituindo normas e recomendações de uso para o desenvolvimento de interfaces para o usuário. Tendo por base as

diretrizes de interface, construímos regras específicas para cada elemento de interface, conforme podemos ver no exemplo da Figura 5. Nesta figura, temos uma regra para o elemento "Bolinha Simples", que é a nomenclatura utilizada na ontologia de elementos de interface que estamos utilizando para o elemento de interface "*radio button*". Esta regra foi construída baseando-se num conjunto de diretrizes de criação de interfaces [21]. Estas diretrizes recomendam que o elemento "Bolinha Simples" deve ser utilizado quando o número de itens a serem exibidos estiver entre "1" e "8", o tipo da variável for "lógica" e o número de escolhas possíveis para o elemento for igual a "1".

```
elemento(NumeroPosicoes, NumeroEscolhas, TipoVariavel, NomeObjeto) :-
        TipoVariavel = logica,
        NumeroPosicoes < 8,
        NumeroPosicoes > 1,
        NumeroEscolhas = 1,
        NomeObjeto = 'Bolinha Simples'.
```

Figura 5. Exemplo de regra criada baseando-se nas diretrizes de criação de interfaces.

É importante ressaltar que, em relação às diretrizes de design de interfaces, estamos nos concentrando nas diretrizes de criação de interfaces da linguagem Java, por ser esta linguagem portável entre diferentes sistemas operacionais. Outro fato a salientar é que estes conjuntos de regras estão representados como regras de produção e estão armazenados na BCDA em um sistema de *frames* [15], sendo ambos implementados na linguagem de programação Prolog.

4.2. O Funcionamento do Sistema Proposto

Na implementação do módulo de geração de interfaces, foram desenvolvidas duas classes de agentes: 1) os agentes de consulta e 2) os agentes construtores de interfaces. Estas duas classes de agentes interagem diretamente com os demais elementos do subsistema de criação de extensões na execução de suas tarefas. Em particular, elas interagirão com o interpretador LPUF [2], que é um agente de software responsável pela interpretação do código da LPUF, pela resolução

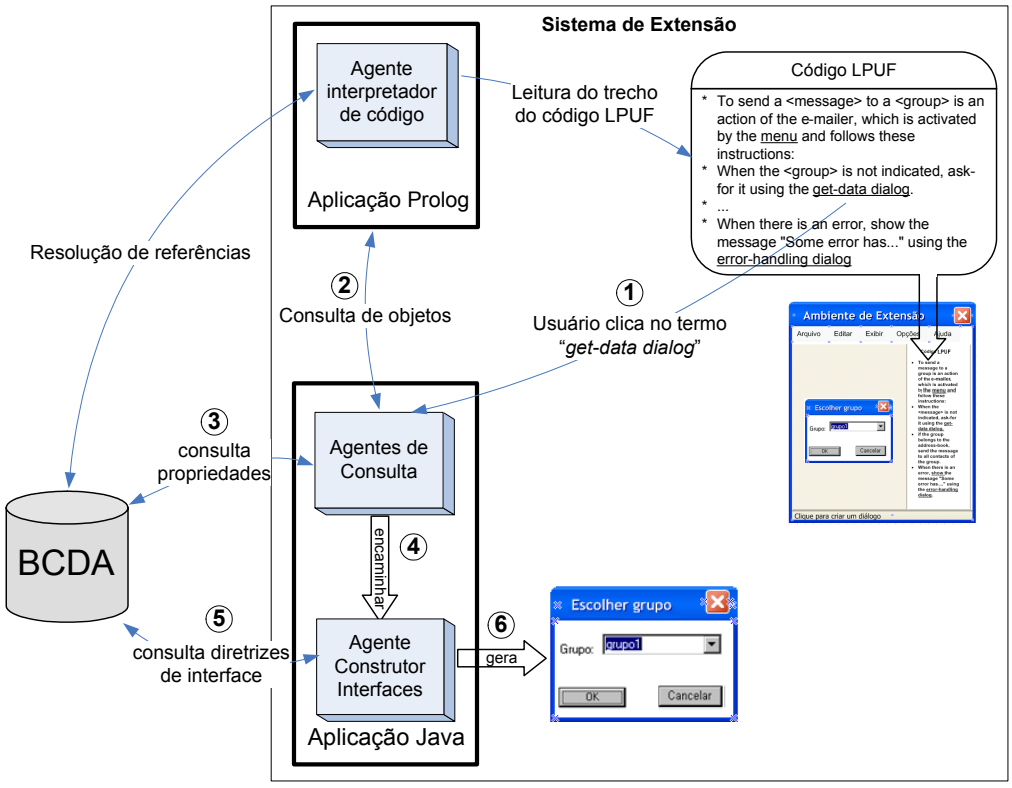


Figura 6. Arquitetura de funcionamento do sistema proposto.

das anáforas e elipses presentes neste código e pela geração do código da extensão. Elas também interagirão com a "Agenda" do sistema que é um agente de software que atua como um assistente para o usuário, sendo responsável pelo gerenciamento do processo de criação de extensões [4] como um todo. A arquitetura de funcionamento dos agentes propostos neste trabalho pode ser visualizada na Figura 6.

É importante salientar que no Modelo Semiótico a tarefa de criação de elementos de interface está intimamente relacionada à criação do código da extensão do usuário. Deste modo, durante a criação do código, o usuário deverá usar elementos da LPUF que permitam criar um *hyperlink* com os agentes criadores de interface, conforme foi descrito na seção 2.2.

Assim, podemos descrever o funcionamento do sistema proposto na geração de diálogos utilizando, como exemplo, o código da Figura 2, apresentado na seção 2.2. Deste modo, inicialmente o usuário, expressando sua intenção de criar um diálogo, clicará no *hyperlink* "*get-data dialog*", que se encontra dentro do código da extensão — passo 1 da Figura 6. Esta ação ativará o agente de consulta que realizará uma primeira consulta ao agente interpretador de código — passo 2 da Figura 6. Desta consulta resultará a sentença na qual se encontra o *hyperlink* dentro do código da extensão. Portanto, o agente construtor de interfaces somente terá acesso aos objetos do texto que estejam anteriormente ao *hyperlink* clicado. Esta sentença será disponibilizada pelo agente interpretador de código na forma de uma árvore hierárquica, a qual será processada pelo agente de consulta por meio de um mecanismo de inferência para a obtenção dos objetos presentes na sentença. A seguir, o agente de consulta realizará uma consulta ao modelo de domínio armazenado na BCDA para a obtenção das propriedades dos objetos retornados — passo 3 da Figura 6. Para isto ele empregará os relacionamentos do objeto com as demais entidades presentes no modelo de domínio.

Em seguida, o conjunto de valores encontrados nas etapas anteriores é repassado ao agente construtor de diálogos — no passo 4 da Figura 6 —, o qual realizará uma consulta no modelo de interação armazenado na BCDA para obter os elementos de interface que se relacionam com os objetos especificados — no passo 5 da Figura 6. Finalmente, o agente construtor de diálogos criará um novo diálogo e adicionará os elementos que foram retornados da consulta às diretrizes de design — no passo 6 da Figura 6.

Após todos esses passos, a interface gerada será exibida para o usuário, que poderá alterar o seu leiaute de acordo com sua preferência. Posteriormente, a interface gerada será armazenada em um arquivo permanente, utilizando para isto uma linguagem de especificação de interface baseada na linguagem XML [8].

Devemos ressaltar que existe um outro processo que também realiza a criação de elementos de interface e faz parte de nosso trabalho, o processo de construção de menus, que por motivo de espaço não vai ser discutido aqui, podendo ser consultado em Jeronymo [11]. Assim, sumarizando a arquitetura de funcionamento do sistema proposto, podemos transformá-la no processo abstrato da Figura 7, apresentada a seguir. Este processo contempla tanto a criação de diálogos quanto a criação de mensagens e está representado por um diagrama de atividades constituindo-se das seguintes etapas:

1. A leitura do trecho de código LPUF, por meio de agentes de software específicos;
2. A consulta na base de conhecimento para obtenção dos valores necessários a criação do diálogo ou mensagem, por meio de agentes de software construtores de interfaces;
3. A geração do diálogo ou da mensagem. Caso a geração seja de um diálogo (3a) e exista um conflito no retorno da consulta, haverá uma outra etapa para a resolução pelo usuário dos conflitos dos elementos de interface a serem usados que será auxiliado pelo agente construtor de interface; caso seja de mensagem (3b), a resolução não será necessária;
4. A alteração do leiaute da interface proposta, caso seja necessário;
5. A aceitação da interface proposta;
6. A transformação da interface gerada em objeto persistente.

Desta forma, na etapa 1 da Figura 7, é realizada a leitura no trecho do código LPUF. Na etapa 2, os agentes consultam a BCDA para a obtenção de informações necessárias à construção do diálogo e, para isso, utilizam-se dos termos obtidos no passo anterior. Na etapa 3, após a definição de qual elemento de interface será gerado, o agente construtor de diálogos abre um ambiente de construção de interface com o leiaute e os elementos propostos onde o usuário poderá aceitar ou alterar a interface, por meio das etapas 5 e 4 respectivamente. Caso seja escolhida a geração de um diálogo e existam conflitos na escolha dos elementos de interface, o usuário é auxiliado pelo agente construtor de diálogos. A última etapa do processo é responsável pelo armazenamento do diálogo gerado em um arquivo XML, que após as verificações de consistência da extensão gerada será armazenada na BCDA, de modo a garantir a consistência dos objetos na base. Devemos lembrar que o usuário poderá a qualquer momento abandonar o processo de construção do diálogo e retomá-lo posteriormente.

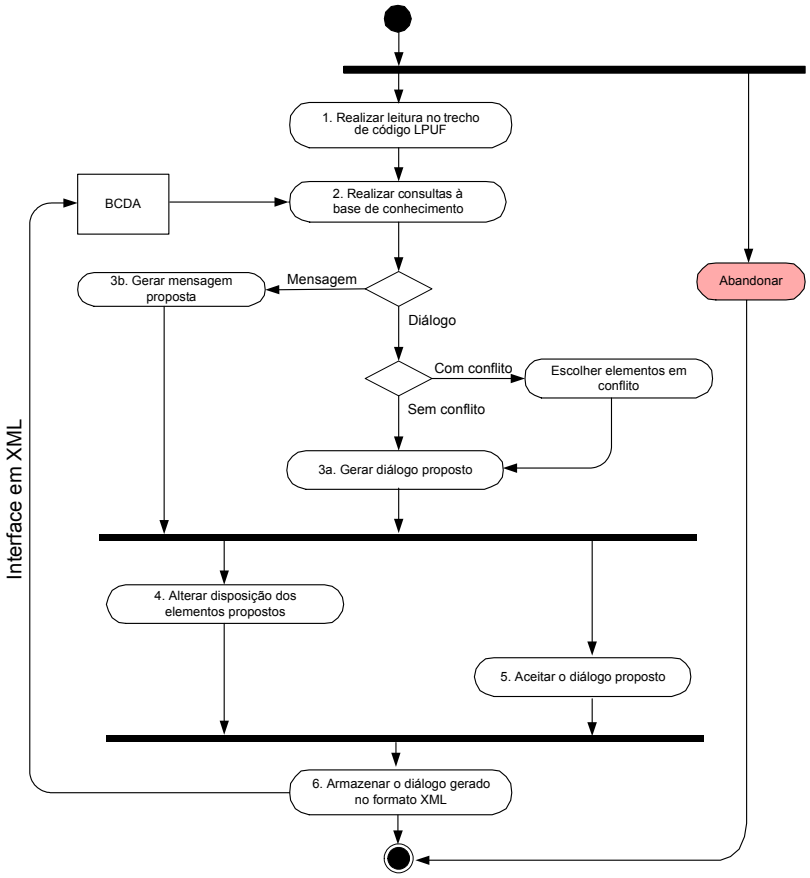


Figura 7. Processo de criação de diálogos de obtenção de dados.

5. CONCLUSÃO

Durante o desenvolvimento deste projeto encontramos poucos trabalhos correlatos. No entanto, diferentemente de nosso projeto, grande parte destes trabalhos utilizava modelos predefinidos de interfaces para a tarefa de geração de interfaces. Em particular, o trabalho que mais se aproximou ao nosso foi o Seguia de Vanderdonckt [24]. Comparado ao nosso trabalho, o Seguia possui grandes semelhanças, principalmente, quanto à utilização de um conjunto de regras que descrevem as diretrizes de criação de interfaces e um conjunto de regras que descrevem os guias de estilo específicos de uma plataforma. Porém, além das regras empregadas no Seguia, possuímos um conjunto de regras definidas pelo próprio designer da aplicação. Estas regras determinam a maneira pela qual os elementos de interface são organizados para criar a linguagem única de interação da aplicação [5] em questão, limitando as possibilidades de criação de interfaces pelo usuário. Esta imposição é necessária para garantir a não degradação das interfaces criadas nas extensões e, por conseguinte, para manutenção dos princípios da Engenharia Semiótica. Entretanto, a principal diferença com o Seguia é que nós estamos preocupados com os usuários finais, enquanto o Seguia foi construído para apoiar programadores. Por esta razão o Seguia utiliza uma ontologia de elementos de interface baseada nas funções que os elementos desempenham na interface para auxiliar na criação de interfaces. As diferentes necessidades de nosso trabalho levaram a criação de uma ontologia de elementos de interface específica para a criação de interfaces por usuários finais. A ontologia resultante foi dividida em três categorias de elementos: elementos de ativação (menus ou atalhos), elementos de diálogo e elementos de mensagens. Estas categorias representam os tipos de interfaces que um usuário final pode criar por meio do sistema proposto.

Um grande diferencial de nosso trabalho, em relação aos ambientes de desenvolvimento de software atuais, está na reversão do processo normal de desenvolvimento de interface com o usuário no software. Nos ambientes atuais, o código e os elementos de interface são desenvolvidos separadamente e depois é feita a ligação entre eles. Em nosso trabalho, partimos diretamente da descrição da funcionalidade da aplicação para a criação dos elementos de interface. Deste modo, o usuário desenvolverá o código de sua extensão na forma textual e quando necessitar da criação de algum

tipo de interface, ele ativará um conjunto de agentes construtores de interface por meio de um *hyperlink* presente no código da LPUF. Estes agentes operam em um ambiente gráfico da linguagem de interação do usuário. Logo, o usuário trabalhará a construção da interface diretamente na interface da aplicação, empregando uma linguagem que ele já conhece, reduzindo, deste modo, a sobrecarga cognitiva associada à criação de extensões no ambientes atuais. Assim, nossa proposta procura reduzir a dificuldade encontrada pelos usuários finais em fazer a ligação entre o código e a interface por ele gerada. A reversão do processo de desenvolvimento de interfaces faz com que o código da interface seja ligado com a funcionalidade automaticamente, não sendo mais necessário ao usuário fazer a ligação a posteriori.

Devemos deixar claro que existem limites na utilização dos resultados deste trabalho. Um desses limites é o fato de que estes resultados não se aplicam diretamente aos sistemas adaptativos, porém, é possível adequá-los para permitir a integração com tais os sistemas. Para tanto, será necessário alterar o processo de utilização dos agentes construtores de interface para que eles sejam ativados pelos agentes geradores de generalizações presentes nas aplicações adaptativas. Estas modificações estão sendo avaliadas em outro trabalho [20] que está sendo desenvolvido dentro do âmbito do projeto APEX. Outro limite refere-se ao fato de que os elementos de interface considerados não levam em conta a técnica de manipulação direta. Esta limitação está relacionada ao uso do Modelo Semiótico como base teórica de nosso trabalho, pois, na sua versão atual, o Modelo Semiótico também apresenta está limitação.

Foram feitos alguns testes preliminares para avaliar as possibilidades de geração de interfaces pelo sistema proposto. Os resultados destes testes nos mostraram como definir claramente a estrutura da base de conhecimento. No entanto, é necessária a realização de mais testes para avaliar o ganho real de usabilidade deste sistema.

Agradecimentos

Os autores gostariam de agradecer a Fundação Araucária que gentilmente concedeu uma bolsa de pesquisa para a realização do projeto APEX, no âmbito do qual este artigo foi desenvolvido.

Referências

[1] Cypher, A. *Watch What I Do: Programming by Demonstration*, The MIT Press, Cambridge, MA. 1993

[2] Da Silva, S. R. P. e Pinheiro, J. M. Um Framework para Criação de Linguagens de Domínio Específico. *Anais VIII SBLP (8th Brazilian Symposium on Programming Languages)*. Universidade Federal Fluminense. UFF. Niterói. RJ. 2004.

[3] Da Silva, S. R. P. *APEX - Aplicações Extensíveis por Usuários Finais*. Projeto aprovado pela Fundação Araucária, 2003.

[4] Da Silva, S. R. P. *Um modelo Semiótico para programação por usuários finais*. Tese de Doutorado. Departamento de Informática. PUC-Rio, 2001.

[5] De Souza, C. S; Barbosa, S. D. J. and Da Silva, S. R. P. Semiotic Engineering Principles for Evaluating End-User Programming Environments. In *Interacting with Computers*. 2001. p. 467-495

[6] De Souza, C. S. The Semiotic Engineering of User Interface Languages. *International Journal of Man- Machine Studies*. Nº 39, 1993. p. 753-773.

[7] Girgensohn, A. *End-user modifiability in knowledge based design environments*. PH.D. Thesis, University of Colorado at Boulder. 1992

[8] Harold E. R. *Processing XML with Java. A Guide to SAX, DOM, JDOM, JAXP, and TrAX*. Addison- Wesley. 2002.

[9] Hypercard — *Apple® HyperCard® Script Language Guide*. Addison Wesley Publishing Company. Apple Computer Inc. 1998.

[10] Jakobson, R. Closing Statements: Linguistics and Poetics. In: SEBEOK, T. (ed.) *Linguistics and Communication*. New York: The MIT Press. 1960.

[11] Jeronymo, M. A. *Geração Automática de Interfaces para Aplicações Extensíveis por Usuários Finais*. Dissertação de Mestrado. Departamento de Informática. UEM- Paraná, 2004.

[12] Kim, W. C. and Foley, J. D. Providing High-level Control and Expert Assistance in the User Interface Presentation Design. *In Human Factors in Computing Systems. Proceedings INTERCHI'93*, Amsterdam, The Netherlands, April, 1993. p. 430-437.

[13] Lieberman, H. (ed.). *Your Wish is my Command: Programming by Demonstration*. Morgan Kaufmann, San Francisco. 2001

[14] Microsoft Corporation. *The Microsoft Windows User Experience*. Microsoft Press International. September 1999.

[15] Minsky, M. A framework to represent knowledge. In *The Psychology of Computer Vision*. New York. McGraw-Hill, 1975.

[16] Myers, B. A. *UIMSs, Toolkits, Interface Builders*. Human Computer Interaction Institute. Carnegie Mellon University. 1996.

[17] MS VBasic — *Microsoft® Visual Basic® .NET Language Reference*. Microsoft Press. Microsoft Corporation. 2002.

[18] Nardi, B. *A Small Matter of Programming*, MIT Press, Cambridge, Ma. 1993.

[19] Olsen, D. R. A Programming Language Basis for User Interface Management. In *Human Factors in Computing Systems. Proceedings SIGCHI'89*, Austin, TX, April, 1989. p. 171-176.

[20] Penteado, R. R. M. *O Uso do Conhecimento Prévio do Domínio na Automatização da Interação do Usuário com o Software*. Dissertação de Mestrado. Universidade Estadual de Maringá. Maringá, PR. 2004.

[21] Sun Microsystems. *Java Look and Feel Design Guidelines*. 2nd Ed.ition. Version 2.0. Sun Microsystems, Inc. February 2001.

[22] Szekely, P.; Luo P. and Neches, R. Beyond Interface Builders: Model-Based Interface Tools. In *Human Factors in Computing Systems. Proceedings of the INTERCHI'93*, Amsterdam, The Netherlands, April, 1993. p. 383-390.

[23] Vanderdonckt, J. and Berquin, P. Towards a Very Large Model-based Approach for User Interface Development, in *Proc. of the 1st Int. Workshop on User Interfaces to Data Intensive Systems UIDIS'99* (Edinburgh, 5-6 September 1999), N.W. Paton & T. Griffiths (eds.), IEEE Computer Society Press, Los Alamitos, 1999. p.76-85.

[24] Vanderdonckt, J. and F. Bodart. Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection, in *Proc. of the ACM Conf. on Human Factors in Computing Systems INTERCHI'93* (Amsterdam, 24-25 April 1993), S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel & T. White (eds.), ACM Press, New York, 1993. p. 424-429.

[25] Zanden, B. V. and Myers, B. A. Automatic, Look-and- Feel Independent Dialog Creation for Graphical User Interfaces. In *Human Factors in Computing Systems. Proceedings of the SIGCHI'90*, Seattle, WA, April, 1990. p. 27-34.

Uma Ferramenta Interativa para Visualização e Exploração de Imagens Médicas

Leandro M. Iglezias, Ricardo Melo Czekster, Isabel Harb Manssour
Pontifícia Universidade Católica do Rio Grande do Sul, Faculdade de Informática,
Av. Ipiranga 6681, Prédio 30, Bloco 4
90619-900 Porto Alegre, RS, Brasil
leeiglezias@viars.com.br, rczekster@gmail.com, manssour@inf.pucrs.br

Abstract

With the evolution of medical image acquisition techniques, e.g. Computed Tomography and Magnetic Resonance Imaging, the capacity and fidelity of image-based diagnosis were extended. The 3D visualization of medical images and the utilization of interactive tools to explore the data volume are very important for the health professionals. Among these tools we can distinguish: measurement extraction (to identify distances, areas and volumes), cutting tools (to allow the visualization of inner structures in the volume), and transfer function (to associate different color and opacity values to the volume structures). This work presents an application that allows the 3D visualization of medical images and provides a new approach for the visualization of inner structures in medical images. Interactive tools for measure extraction, cutting and transfer function application are available. Tests with users and images of the magnetic resonance of one phantom were used to increase system accuracy.

Keywords: Visualization, Medical Imaging, Measurement Extraction, Cutting Tools, Transfer Function.

Resumo

Com o aperfeiçoamento de técnicas de aquisição de imagens médicas, como a Tomografia Computadorizada e a Ressonância Magnética, a capacidade e a fidelidade do diagnóstico por imagens foram ampliadas. A visualização 3D de imagens médicas e o uso de ferramentas interativas para explorar o volume de dados são muito importantes para os profissionais da saúde. Entre estas ferramentas destacam-se: a extração de medidas (para mensurar distâncias, áreas e volumes), ferramentas de corte (para possibilitar a visualização de estruturas internas no volume) e funções de transferência (para associar diferentes valores de cor e opacidade às estruturas do volume). Este trabalho apresenta uma aplicação que permite a visualização 3D de imagens médicas e provê uma nova abordagem para a visualização de estruturas internas em imagens médicas fundamentada na utilização de ferramentas de corte e funções de transferência. Ferramentas interativas de extração de medidas, de corte e de aplicação de funções de transferência foram disponibilizadas. Testes com usuários e imagens de ressonância magnética de um *phantom* foram usadas para teste da precisão do sistema.

Palavras chaves: Visualização, Imagens Médicas, Extração de Medidas, Ferramentas de Corte, Funções de Transferência.

1 INTRODUÇÃO

Com a evolução das tecnologias de aquisição de imagens em termos de resolução e distinção de tecidos, a capacidade e a fidelidade do diagnóstico por imagens foram ampliadas. Diversas modalidades de aquisição de imagens, tais como Tomografia Computadorizada (CT) e Ressonância Magnética (MRI), vêm sendo utilizadas com o intuito de facilitar os diagnósticos médicos.

A Visualização Volumétrica denota o conjunto de técnicas utilizadas na apresentação de dados científicos 3D. Um dos seus principais objetivos é permitir a identificação e exploração de estruturas internas no volume. Como o volume é normalmente um "bloco de dados", não se pode visualizar o seu interior, a menos que se assuma que é possível ver através de *voxels* transparentes, ou que é possível remover *voxels* que estão

na frente da região de interesse (ROI ou *Region of Interest*), o que pode ser feito através de técnicas de corte. No caso de dados médicos, devido às várias modalidades de aquisição existentes, é possível identificar na literatura diversos esforços destinados a explorar maneiras diferentes de visualizar e interagir com dados volumétricos [1, 3, 2]. Portanto, nos últimos anos, novas técnicas de visualização e interação com estes dados foram sendo desenvolvidas.

Considerando que o objetivo dos médicos normalmente é visualizar e quantificar isoladamente estruturas dentro de um volume, faz-se necessário prover ferramentas interativas para isolar, visualizar e extrair medidas destas estruturas. Atualmente o Centro de Diagnósticos por Imagens (CDI) do Hospital São Lucas da PUCRS carece de ferramentas com interfaces amigáveis para este propósito, pois os sistemas disponíveis normalmente representam grandes investimentos e invariavelmente possuem interfaces muito complexas.

O principal objetivo deste trabalho é apresentar o ExploMed, um sistema interativo de visualização que fornece ferramentas para aferição de distâncias, ângulos, perímetros e áreas, além de possibilitar a visualização 3D de imagens médicas e a aplicação de ferramentas de corte e funções de transferência. Esta aplicação, que é uma extensão de um trabalho apresentado anteriormente, o MedScape [5], foi desenvolvida utilizando o *Visualization Toolkit* (VTK) [4] e o *Borland C++ Builder*. As principais alterações em relação ao MedScape [5] são: inclusão das ferramentas de corte, possibilidade de manipulação interativa das funções de transferência e aperfeiçoamento da interface gráfica com o usuário. Para validação do sistema foram feitos testes de avaliação de usabilidade com potenciais usuários e foi utilizado um *phantom*, que consiste em um dispositivo destinado a testes e calibragem dos equipamentos de aquisição.

Este artigo está organizado da seguinte maneira. A Seção 2 apresenta alguns trabalhos relacionados. A implementação e as ferramentas interativas disponíveis no sistema são descritas na Seção 3. As conclusões e trabalhos futuros são apresentados na Seção 4.

2 TRABALHOS RELACIONADOS

Muitos sistemas de visualização de imagens médicas já foram desenvolvidos e suas descrições podem ser encontradas na literatura. Alguns são específicos para uma aplicação e disponibilizam um determinado conjunto de ferramentas. Outros podem ser utilizados com diferentes volumes de dados, tais como dados resultantes de simulações. Um problema é que a maioria destes programas possui um custo muito elevado, dificultando a sua aquisição.

Um sistema comercial que foi desenvolvido utilizando o VTK, O VolView [13], é um exemplo de sistema interativo para visualização e exploração de dados volumétricos. Apesar de disponibilizar muitas funcionalidades, incluindo ferramentas de corte e de manipulação de funções de transferência, não permite a extração de medidas e apenas um plano oblíquo pode ser usado para realizar o corte.

Já o Osiris [14], que é gratuito, foi projetado para ser uma ferramenta genérica de manipulação de imagens médicas. Suas principais vantagens são a presença de filtros e recursos de análises quantitativas. Permite a seleção de ROIs e VOIs (*Volume of Interest*, ou Volume de Interesse), não possui finalidades comerciais e está disponível para diversas plataformas. Entretanto, não contém muitas ferramentas para realizar cortes no volume de dados.

Similar ao ExploMed, o VolVis foi projetado para permitir que programadores adicionassem novas representações e algoritmos [15]. Também está disponível gratuitamente, mas não possui ferramentas para extração de medidas. Outro sistema gratuito que disponibiliza uma série de ferramentas para extração de medidas é o ImageJ [16]. Desenvolvido em Java, além de ser portável, pode ser estendido através da inclusão de *plugins*.

3 DESCRIÇÃO DO SISTEMA

Vários requisitos foram considerados para o desenvolvimento do sistema, tais como, possibilidade de extensão e desenvolvimento de ferramentas interativas fáceis de manipular e úteis para o usuário final. A Seção 3.1 apresenta o ambiente de desenvolvimento e o *toolkit* utilizado. A modelagem e a interface do sistema são descritas, respectivamente, nas Seções 3.2 e 3.3. A Seção 3.4 apresenta as ferramentas de interação desenvolvidas, e o processo de validação das mesmas é descrito na Seção 3.5. Uma avaliação do sistema, comparando com outros já existentes na literatura, é feita na Seção 3.6.

3.1 Ambiente de Desenvolvimento

Considerando os requisitos para implementação do ExploMed, de ser um sistema flexível, portável e fácil de estender, optou-se pela utilização da linguagem de programação C++ padrão (ANSI) e do *toolkit* VTK [4]. Apenas para o desenvolvimento da interface foi utilizado o ambiente de desenvolvimento *Borland C++ Builder*, o que fez com que o sistema possa ser executado apenas no ambiente *Microsoft Windows*.

O VTK é um *toolkit* disponibilizado gratuitamente pela Kitware Inc.[7] possuindo código fonte aberto e totalmente portável. Basicamente, o VTK consiste em uma biblioteca de classes implementadas em C++ e utilizada para o processamento de imagens e visualização científica, que possui diversos níveis de interface para linguagens interpretadas, incluindo Tcl/Tk, Java e Phyton.

O VTK possui um enorme conjunto de classes que implementam diversas funcionalidades. Alguns exemplos são as classes *vtkActor2D*, que corresponde a um ator que exibe dados 2D, e *vtkActor2DCollection*, utilizada para definir uma lista de atores. *vtkImageViewer* é uma classe destinada a exibir dados 2D. *vtkImageReslice* é uma classe auxiliar utilizada para obter os planos ortogonais ou oblíquos de um volume. A classe *vtkRenderWindow* é utilizada para especificar o comportamento de uma janela de *rendering*, provendo, por exemplo, métodos para definir o seu tamanho e o número de *bits* por *pixel*. Suas instâncias são independentes de plataforma. A classe responsável por controlar as luzes presentes na cena, as câmeras e os atores, tendo como principal objetivo a geração da imagem, é *vtkRenderer*. Objetos da classe *vtkRenderWindowInteractor* gerenciam o mecanismo de interação (*mouse*, teclado). Os algoritmos para visualização volumétrica e aplicação de funções de transferência, são disponibilizados nas classes *vtkVolumeRayCastFunction* e *vtkColorTransferFunction*. Através das classes *vtkBoxWidget*, *vtkPlaneWidget* e *vtkImplicitPlaneWidget* foi possível incluir e manipular planos e volumes de corte.

3.2 Modelagem

Nesta seção é descrita a arquitetura do ExploMed, que é baseada no padrão *Model-View-Controller* (MVC) [19], e apresentada neste trabalho usando a *Unified Modeling Language* (UML) [20]. MVC consiste em uma tríade de classes extremamente usadas em sistemas interativos para construção de interfaces com o usuário. A implementação do padrão MVC mantém o núcleo funcional do sistema independente desta interface, que geralmente está sujeita a alterações e adaptações. Por este motivo, é essencial o desenvolvimento de uma arquitetura que dê suporte às alterações da interface sem causar efeitos colaterais nas funções específicas da aplicação ou do modelo [19].

A Figura 1 mostra uma descrição simplificada do modelo conceitual em UML, que foi projetado de forma a tornar independentes os objetos responsáveis pelo gerenciamento da interface. Algumas classes do VTK foram incluídas, e iniciam com *vtk*, com exceção das classes *vtkCallBackBoxWidget* e *vtkCallBackPlanoCorte*, que foram desenvolvidas como especializações da classe *vtkCommand*. As classes de controle, estendidas da primeira versão do sistema, o MedScape [5], iniciam com as letras *msc* e as classes de interface inicial com *frm*. Para simplificar o diagrama apresentado na Figura 1, algumas classes básicas não foram incluídas.

Assim, de acordo com o padrão MVC, o modelo corresponde a algumas classes do VTK em conjunto com outras desenvolvidas especificamente para a aplicação em questão. O controlador é responsável pela interação do modelo com os dados, controlando eventos e garantindo que toda vez que os dados são modificados, estes são reapresentados ao usuário. Além disso, desenvolvendo uma nova interface, o sistema pode ser facilmente adaptado para executar em outra plataforma (exemplo: Linux).

Uma das mais importantes classes é a *mscView*, que encapsula as classes fundamentais do *pipeline* de visualização do VTK, como *vtkRenderWindow*, *vtkRenderWindowInteractor* e *vtkRenderer*. Como a visualização de dados médicos usualmente necessita de mais de uma vista, um objeto *mscView* pode corresponder a uma vista axial, coronal, sagital ou 3D para o órgão reconstruído. Assim, cria-se uma dependência entre objetos de um-para-muitos, e quando um objeto (neste caso, o modelo) altera seu estado, todos os seus dependentes devem ser notificados e atualizados automaticamente [21]. A classe *mscInteractorStyle* é uma extensão de *vtkInteractorStyleUser* e é responsável pelo controle da interação com o usuário, definindo alguns métodos para capturar eventos do *mouse*.

Outra classe fundamental ao sistema é a *mscVolume*, pois é responsável pela abertura e armazenamento de um volume de dados médicos. Este objeto possui como atributo uma instância da classe *mscVolumeProperties*, que é usada para guardar as diversas propriedades sobre o volume, tal como suas dimensões e seu número de fatias. Um objeto *mscVolume* também possui como um atributo um objeto para ler o volume. Os métodos necessários para a conversão de dados do formato DICOM para RAW, que são os dois formatos

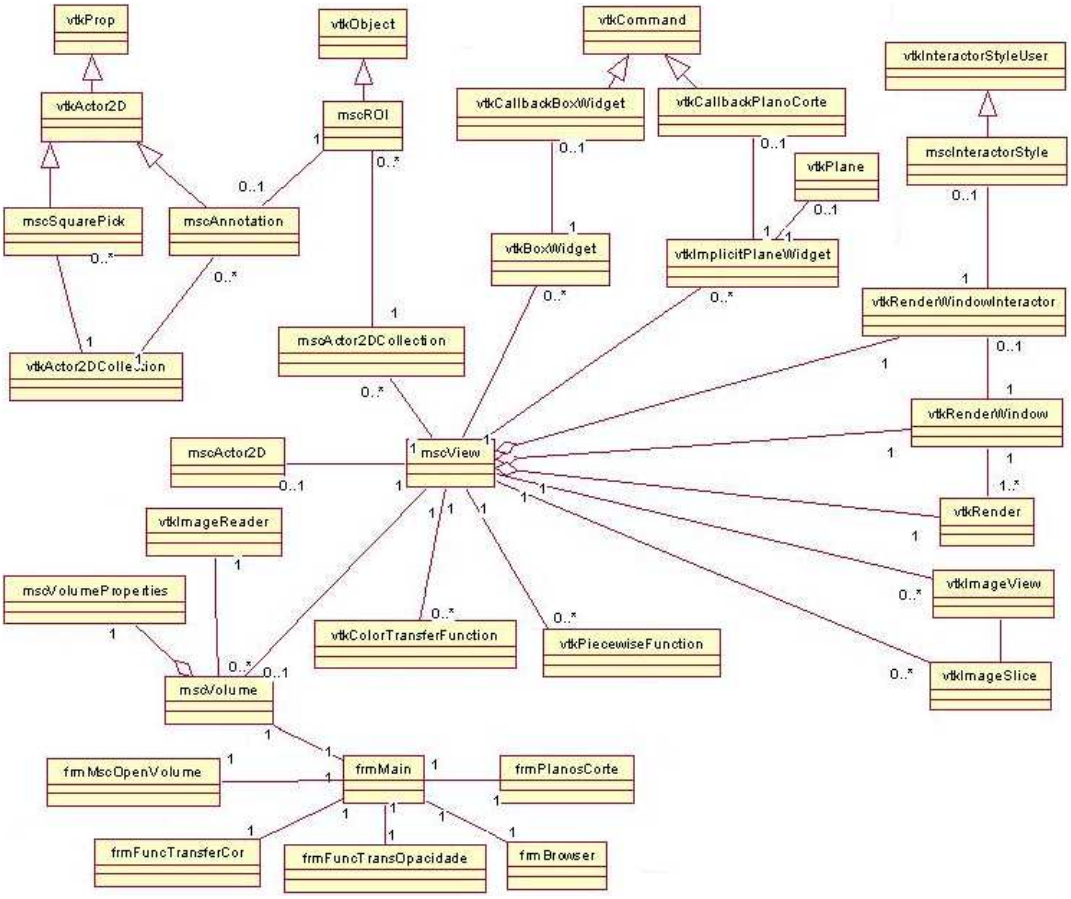


Figura 1: Diagrama de classes do ExploMed, incluindo as classes desenvolvidas no MedScape [5])

aceitos no ExploMed, estão associados à classe *mscConversion*, que não possui conexão direta com qualquer outra classe. O programa de linha de comando *dicom2* [22] é usado para converter os dados.

A classe *frmMain* disponibiliza a interface principal com o usuário, da qual é possível acessar todas as funcionalidades do sistema através de botões ou menus. Para a maioria das opções que podem ser selecionadas, uma janela é aberta, e para cada janela existe uma classe que disponibiliza sua interface gráfica. Por exemplo: a classe *frmBrowser* provê uma interface para navegar entre as fatias ortogonais e escolher qual das quatro janelas de visualização deseja manter aberta (vistas axial, coronal, sagital ou 3D); *frmFuncTransferCor* e *frmFuncTransOpacidade* são as classes que disponibilizam uma interface gráfica para inserir ou remover pontos que formam, respectivamente, as funções de transferência de cor e opacidade que podem ser aplicadas na visualização volumétrica.

3.3 Interface do Sistema

Como um dos objetivos é gerar ferramentas úteis para profissionais que não estão acostumados com a utilização deste tipo de aplicação, a interface foi projetada de maneira a evitar que o usuário informe ao sistema em um único passo, os vários parâmetros numéricos necessários para gerar uma imagem. Nesta seção é sucintamente descrita a interface do ExploMed, incluindo sua utilização e suas principais funcionalidades, muitas das quais acessíveis também através de teclas de atalho. A Figura 2 apresenta a interface completa, que é composta das seguintes janelas:

- Janela Principal (a): contém todas as funcionalidades do sistema através de botões ou menus;
- Janela Navegação (b): utilizada para selecionar qual das quatro janelas de visualização (axial, coronal,

sagital e 3D) que ser quer visualizar, podendo também selecionar qual fatia será visualizada nas vistas ortogonais;

- Janela de Propriedades (c): permite configurar o contraste e o brilho da imagem para a visualização dos planos ortogonais;
- Janela de Ferramentas de Corte (d): possui opções para cortes no volume com até 6 planos simultâneos e/ou corte através de um cubo por inclusão ou por exclusão;
- Janela Abrir (e): utilizada para abrir um arquivo de volume de dados;
- Janelas de Visualização (f): nas quais são exibidas as imagens dos planos axial, coronal e sagital, e da visualização 3D;
- Janela de Função de Transferência de Cor (g): permite alterar a função de transferência de cor que será aplicada à visualização 3D;
- Janela de Função de Transferência de Opacidade (h): permite alterar a função de transferência de opacidade que será aplicada à visualização 3D.

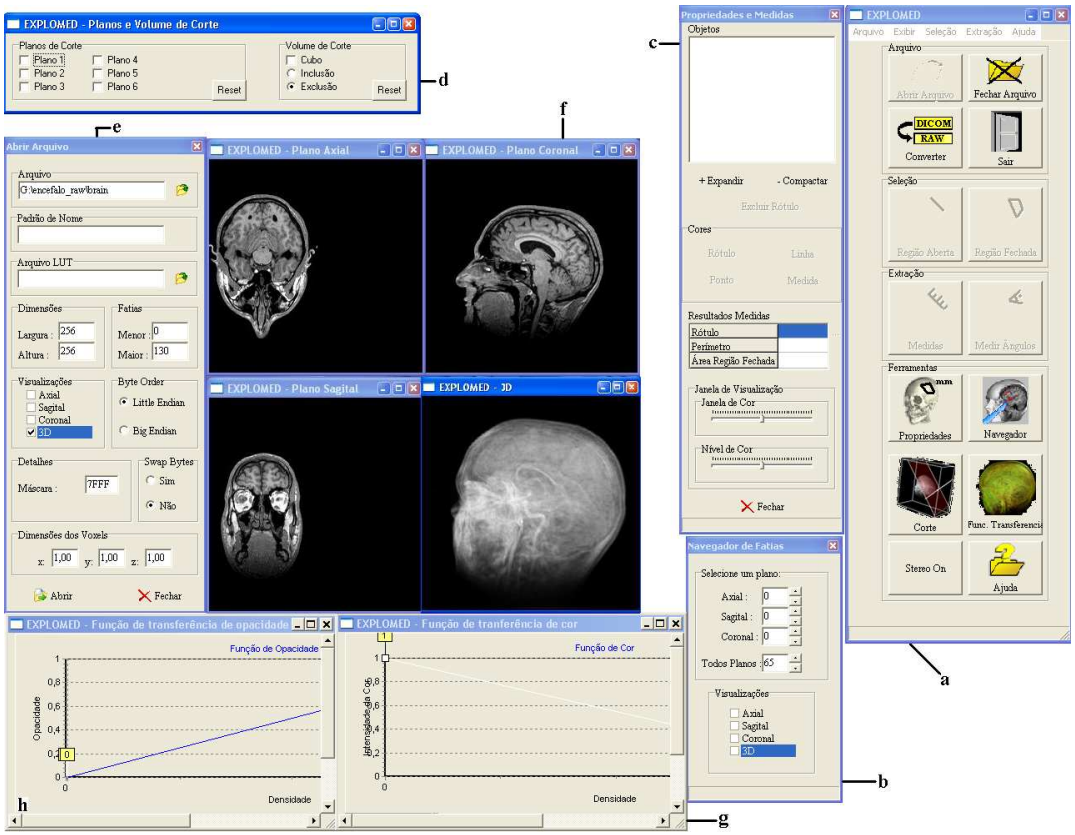


Figura 2: Interface do ExploMed

A execução das diferentes funcionalidades do sistema é ativada quando o usuário clica com o *mouse* sobre um dos ícones da janela principal (Figura 2a). A opção para abrir um arquivo está disponível no menu ou no botão Arquivo. Também tem um botão para selecionar a opção de converter arquivos DICOM para RAW. Após abrir um volume, é aberta uma janela na qual o usuário deve informar alguns parâmetros da aquisição, tais como tamanho do *voxel*, dimensões do volume e ordem do *byte* (Figura 2e). Depois, é possível percorrer as "fatias", bastando para isto clicar no botão Navegador (Figura 2a). Na janela aberta (Figura

2b) o usuário poderá selecionar a fatia que deseja visualizar em cada plano. Além disso, também é possível abrir ou fechar as janelas de visualização.

Para selecionar a inserção de uma ROI aberta ou fechada, e selecionar entre medir distâncias ou ângulos em cada fatia, é necessário clicar nos botões da janela principal e depois inserir os pontos diretamente na janela de visualização da fatia. Para visualizar a medida de perímetros e áreas, para trocar a cor do rótulo, para selecionar as diferentes ROIs inseridas e para ajustar as cores da imagem dos planos ortogonais, é necessário abrir a janela de propriedades, clicando no botão de propriedades da janela principal.

A manipulação do volume reconstruído na janela 3D é possível apenas através da utilização do *mouse*. Para realizar uma rotação é preciso clicar e segurar o botão esquerdo sobre o objeto, arrastando o mouse para as direções desejadas. A rotação será realizada de acordo com os movimentos. Para realizar *zoom-out* basta pressionar o botão direito sobre o objeto e arrastar para baixo. Arrastando para cima será realizado o *zoom-in*. Para mover o objeto de lugar é necessário pressionar o botão do meio e movimentar o objeto para outra posição. Para soltar o objeto é preciso soltar o botão do *mouse*. Caso este não possua 3 botões, a tecla "*shift*" deve ser pressionada simultaneamente com o botão esquerdo do *mouse*.

Clicando no botão de ferramentas de corte na janela principal, é aberta a janela de ferramentas de corte (Figura 2d), na qual deve ser selecionado o plano que se deseja utilizar. Podem ser utilizados até 6 planos de corte que podem ser manipulados interativamente. Para realizar corte por volume é necessário selecionar a opção de volume e o tipo de corte desejado, por inclusão ou por exclusão. Neste tipo de corte também é possível alterar a posição e o tamanho do volume interativamente.

Para utilizar as funções de transferência, deve-se clicar no botão correspondente na janela principal (Figura 2a). Neste momento, duas janelas serão abertas, uma para função de opacidade (Figura 2h) e outra para função de cor (Figura 2g). A partir daí, basta selecionar a densidade desejada e clicar em alterar para inserir os novos valores. Os gráficos são automaticamente atualizados e somente quando a função estiver pronta o usuário deve clicar no botão aplicar, pois através deste é que a visualização 3D será atualizada.

3.4 Ferramentas de Interação

Até o momento a aplicação desenvolvida possui as seguintes ferramentas de interação: extração de medidas, planos de corte oblíquos, volume de corte e manipulação das funções de transferência. Além disso, exibe todos os planos ortogonais em janelas diferentes, permite fazer o *zoom* e manipular o volume reconstruído para visualizá-lo de diferentes ângulos.

As ferramentas de extração de medidas disponíveis incluem: distância entre dois pontos; seleção de ROIs utilizando-se "polilinhas" para o cálculo de áreas e perímetros; e ângulo entre duas linhas especificadas pelo usuário. Todas estas ferramentas estão associadas a uma "fatia" específica. Partindo do pressuposto que o tamanho do *voxel* é conhecido, foram utilizados alguns conceitos básicos para processar as mensurações. Para calcular a distância entre dois pontos foi utilizado o teorema de Pitágoras apresentado na Equação 1. Neste caso, d corresponde à distância de interesse.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

Uma ROI é definida por uma série de linhas conectadas. Assim, o perímetro pode ser calculado através da soma de todas as distâncias entre os vértices de cada linha da ROI. Para o cálculo da área foi utilizada uma fórmula popular da geometria analítica [18]. Esta fórmula é apresentada na Equação 2, onde a e b correspondem aos pontos do polígono inserido na direção anti-horária. A Figura 3a ilustra uma ROI definida pelo usuário incluindo seu perímetro e sua área.

$$a = \frac{1}{2} * \begin{bmatrix} a_1 & a_2 & \dots & a_n & a_1 \\ b_1 & b_2 & \dots & b_n & b_1 \end{bmatrix} \tag{2}$$

Para o cálculo do ângulo entre duas linhas consideram-se três pontos não paralelos e não perpendiculares. Em geral, estes pontos correspondem aos vértices das linhas. A Figura 3b mostra um exemplo do cálculo de um ângulo, que é processado pela Equação 3, onde m_1 e m_2 consistem nas inclinações das retas.

$$\alpha = \left| \frac{m_1 - m_2}{1 + m_1 m_2} \right| \tag{3}$$

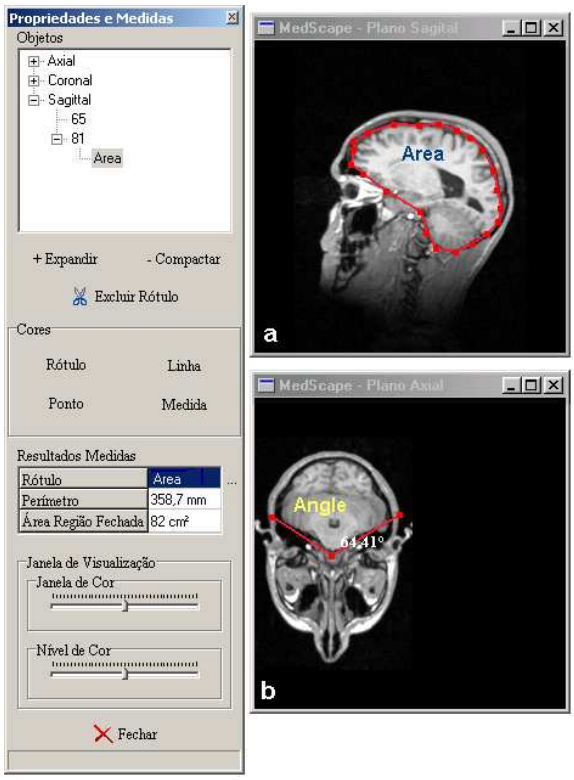


Figura 3: Cálculo de área e perímetro da ROI (a) e do ângulo entre duas retas (b)

Além dos três planos ortogonais (Figura 2), o ExploMed permite incluir e manipular interativamente na janela de visualização 3D, de um até seis planos de corte oblíquos. Para isto, basta selecionar o número de planos desejados na janela de ferramentas de corte (Figura 2d) para que estes apareçam na janela de visualização 3D. Dois exemplos, com um e com seis planos de corte, são apresentados na Figura 4. Para facilitar a interação foram incluídas as *widgets* disponíveis no VTK, possibilitando que o usuário altere a posição e orientação do plano. Clicando sobre o plano é possível colocá-lo na posição desejada, e clicando sobre uma das "setas" que se encontram nos planos é possível movimentá-la, mudando, assim, a inclinação do plano. As setas também indicam de qual lado do plano está sendo feito o corte, conforme ilustra a Figura 4.

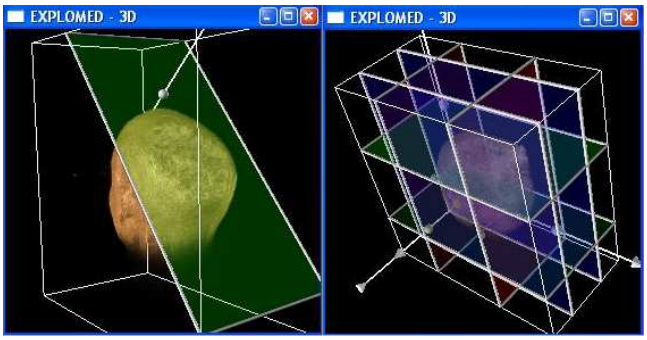


Figura 4: Um e seis planos de corte que podem ser manipulados interativamente.

O corte por volume também foi implementado, e pode ser realizado por inclusão (o que estiver dentro do volume é o que será visualizado) ou por exclusão (o que estiver fora do volume é o que será visualizado). Para

este tipo de corte também foram utilizadas *widgets*, porém, neste caso, para a manipulação do volume de corte. As *widgets* facilitam a manipulação, pois permitem que o usuário clique sobre as "bolinhas" presentes em cada face do objeto, que no caso do ExploMed é um paralelepípedo, e arraste-as para facilmente aumentar ou diminuir o tamanho volume de corte. A seleção de corte por volume é feita na janela de ferramentas de corte (Figura 2d), quando, então, um cubo aparece junto do volume de dados na janela de visualização 3D, como ilustra a Figura 5. Depois, o usuário pode optar pelo corte por inclusão, exemplificado nas duas janelas inferiores da Figura 5, e por exclusão, como mostram as duas janelas superiores.

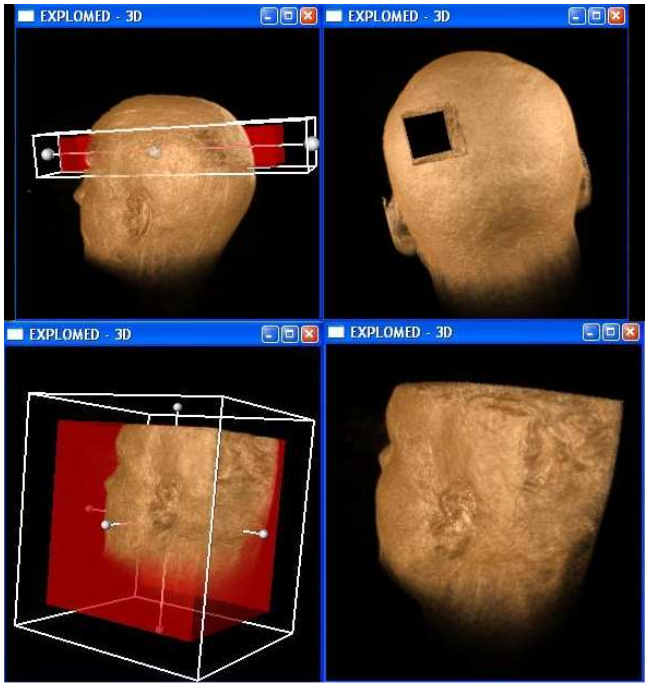


Figura 5: Um volume de corte que pode ser manipulado interativamente, e alternadamente para corte por inclusão e por exclusão.

Por *default* as funções de transferência de cor e opacidade são aplicadas através de interpolação linear. Porém, o usuário tem a alternativa de escolher um arquivo que contenha os parâmetros para as funções de transferência (arquivo .LUT) no momento que é feita a leitura do volume de dados, ou de criar suas próprias funções de transferência através da interface desenvolvida para o sistema. Neste caso, o usuário pode inserir e remover pontos e editar os seus valores. A Figura 6 ilustra a edição da função de transferência de opacidade (a) e de cor (b). Através das caixas de texto o usuário pode incluir o valor de densidade no qual deve ser inserido um ponto e o seu respectivo valor de cor e opacidade.

3.5 Validação e Teste de Usabilidade

Para a validação das ferramentas de corte foram utilizadas imagens adquiridas de um *phantom*, um objeto utilizado para monitorar o desempenho de um equipamento de MRI. O *phantom* disponível no CDI do Hospital São Lucas da PUCRS tem 170mm de diâmetro interno e diversas estruturas internas (Figura 7a), conforme especificado no seu manual. Conforme ilustra a Figura 7b, para aquisição das imagens o *phantom* é inserido em um tipo de invólucro. As imagens MRI deste *phantom* foram visualizadas no sistema, com o intuito de validá-lo, uma vez que as suas dimensões são conhecidas. A Figura 7c mostra que o diâmetro calculado coincide com o especificado no manual. Assim, observa-se que os resultados obtidos pelas ferramentas de extração de medidas estão corretos.

Para as demais ferramentas de interação, responsáveis pela aplicação de volume e planos de corte, além da manipulação de funções de transferência, foi avaliada a qualidade da interação dos usuários com a interface desenvolvida. Foram elaborados *checklists* e formulários de avaliação heurística, que foram aplicados para

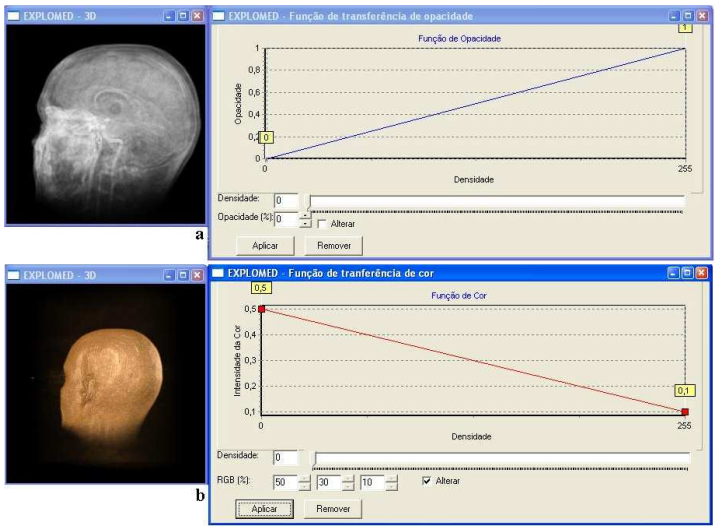


Figura 6: Um volume de corte que pode ser manipulado interativamente, e alternadamente para corte por inclusão e por exclusão.

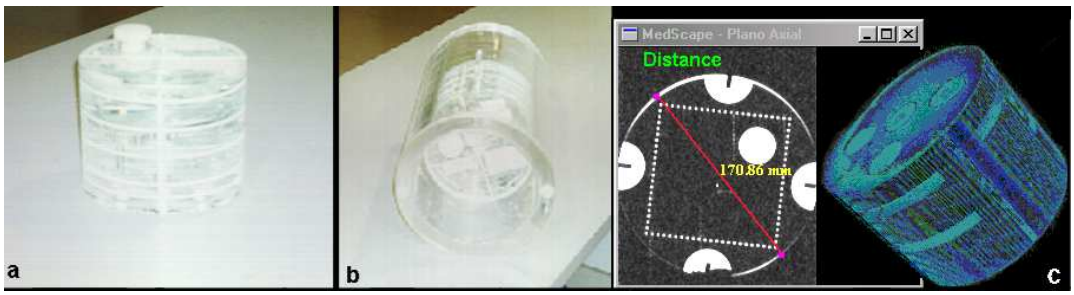


Figura 7: O *phantom* utilizado para validação do sistema (a) e (b), e o diâmetro calculado (c)

um grupo de usuários [8, 9, 10, 11], incluindo físicos-médicos, especialistas na área de interação homem computador e usuários leigos. Todos os usuários executaram uma lista de tarefas (anexo C) antes de preencherem os formulários. Uma análise detalhada dos resultados obtidos permitiu avaliar e aperfeiçoar a interface do sistema.

Da avaliação, pode-se observar que a interface do sistema: permite que o usuário facilmente encontre os dados; apresenta consistência entre as várias telas; utiliza um vocabulário, siglas e abreviaturas fáceis de entender; possibilita uma fácil manipulação dos planos e do volume de corte. Por outro lado, alguns usuários sentiram falta da inclusão de alguns elementos na interface do sistema, acharam que alguns ícones poderiam ser mais intuitivos e sentiram dificuldade na manipulação das funções de transferência. Apesar da existência de alguns pontos fracos, no geral, pode-se afirmar que o sistema foi muito bem avaliado quanto à sua usabilidade [12].

3.6 Avaliação

O sistema desenvolvido provê a inserção interativa de ROIs em qualquer fatia, sem perda de informações quando o usuário visualiza e manipula o volume. Também permite a inclusão de até seis planos de corte simultaneamente ou de um volume de corte, além de manipular as funções de transferência. A possibilidade de exibir apenas as janelas de interesse (3D ou planos ortogonais) é outra funcionalidade do ExploMed. Em compensação, uma das desvantagens é a impossibilidade de abrir outros formatos de dados, tal como DICOM [17].

Comparando a outros sistemas disponíveis atualmente, o ExploMed disponibiliza o conjunto de ferramen-

tas mais utilizadas pelos médicos para auxiliar no diagnóstico. É gratuito, foi validado, possui uma interface intuitiva e foi modelado seguindo o padrão MVC, sendo fácil de portar e estender. Com o desenvolvimento de novas funcionalidades, o ExploMed poderá ser uma ótima alternativa para o VolView, que também faz uso do VTK, mas possui um custo muito elevado.

O Osiris (Seção 2) possui muitas funcionalidades semelhantes ao ExploMed. Uma das suas características, ainda não disponível no ExploMed, é a possibilidade de salvar em arquivo as informações das ROIs. Entretanto, a alternativa de utilizar diferentes ferramentas de corte para visualizar estruturas de interesse e de associar e eliminar ROIs a cada fatia, são inovações do ExploMed.

Alguns sistemas analisados e apresentados na Seção 2 possuem ferramentas para extração de medidas, para realização de cortes no volume e para visualização interativa, porém, nenhum deles permite a inserção interativa de ROIs em qualquer dos planos ortogonais, armazenando estas informações para consultas futuras e alterações de propriedades, uma característica interessante do ExploMed. A possibilidade de usar até seis planos de corte simultaneamente ou um volume para corte por inclusão ou por exclusão também são ferramentas úteis para este tipo de aplicação.

4 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi apresentada uma aplicação interativa para extração de medidas e visualização de estruturas internas utilizando ferramentas de corte em imagens médicas, chamada ExploMed. A arquitetura do sistema desenvolvido possibilita a adição de novas classes, permitindo uma rápida integração de outras funcionalidades. Como todas as classes básicas e fundamentais já foram implementadas, agora se está trabalhando no desenvolvimento e na adaptação de novas ferramentas e algoritmos necessários para melhorar as funcionalidades existentes.

O ExploMed atualmente permite a inserção interativa de ROIs, possibilitando o cálculo de perímetros, distâncias, áreas e ângulos em tempo real. Os resultados das medidas processadas foram validados utilizando-se dados provenientes do *phantom*, sendo que o diâmetro calculado pelo sistema é exatamente o diâmetro real do dispositivo. Foram utilizados cálculos simples para extração de medidas, alcançando excelentes resultados.

Além disso, o ExploMed é gratuito e oferece uma série de possibilidades de realizar cortes no volume de dados para permitir uma análise das suas estruturas internas. É possível incluir até seis planos de cortes que podem ser manipulados interativamente. Através de uma nova abordagem [6], um paralelepípedo também pode ser usado para fazer corte por inclusão ou por exclusão. Estas funcionalidades disponíveis no ExploMed não foram encontradas nas ferramentas similares gratuitas apresentadas na literatura, mostrando seu grande potencial. A atribuição de cor e opacidade aos diferentes tecidos é facilitada pela manipulação interativa das funções de transferência. Através de testes com potenciais usuários se obteve uma resposta positiva no sentido de que foi possível verificar que o sistema é fácil de utilizar e possui as funcionalidades necessárias para auxiliar no diagnóstico médico.

A possibilidade de reuso do software foi levada em consideração durante o projeto do sistema, de maneira que o núcleo funcional do ExploMed é independente de plataforma. Foram aproveitadas ferramentas disponíveis (VTK), ao mesmo tempo em que foi modelado um sistema facilmente extensível. Sistemas de visualização interativos devem ser modulares, facilmente escaláveis e portáveis para diferentes plataformas. Com a arquitetura proposta e as ferramentas implementadas, é possível usar algoritmos otimizados já desenvolvidos, permitindo integrá-los com outras ferramentas de extração de medidas sem ter que reconstruir todos os módulos.

Com o objetivo de incluir ferramentas de segmentação e disponibilizar novas funcionalidades ao ExploMed que permitam o cálculo de volumes, está sendo estudado o *National Library of Medicine Insight Segmentation and Registration Toolkit* (ITK) [7]. Futuros trabalhos também incluem a possibilidade de salvar as informações sobre as ROIs de um volume e o cálculo de distâncias em 3D. Para isto, será necessário associar um objeto 3D com um algoritmo de visualização que identifique a região que está sendo analisada e permita o cálculo de VOIs, como exemplificado por [3]. Além disso, pretende-se fazer uma nova versão da interface utilizando FLTK [23] para tornar o ExploMed independente de plataforma.

Agradecimentos

Este trabalho foi parcialmente financiado pela FAPERGS e PUCRS. Um agradecimento especial ao Maurício Anes pela atenção e por ter fornecido as imagens utilizadas no sistema.

Referencias

[1] B. Lichtenbelt, R. Crane, S. Naqvi. *Introduction to Volume Rendering*. Upper Saddle River, NJ: Prentice Hall, 1998.

[2] D. Weiskopf, K. Engel, T. Ertl. Interactive Clipping Techniques for Texture-Based Volume Visualization and Volume Shading. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, No. 3, (Sept. 2003).

[3] B. Preim, C. Tietjen, W. Spindler, H.-O. Peitgen. Integration of Measurement Tools in Medical 3D Visualizations. *Proceedings of IEEE Visualization*. 27 Oct.-1 Nov. 2002. pp. 21-28.

[4] L.S. Avila et. al. *The VTK user's guide: updated for VTK version 4.2*. Upper Saddle River, NJ: Kitware, 2003.

[5] M.M de Carvalho, R.M. Czekster, I.H. Manssour. Uma Ferramenta Interativa para Visualização e Extração de Medidas em Imagens Médicas. *Revista Eletrônica de Iniciação Científica*, Vol. 3, No. 3, (2003). pp. 1-10.

[6] I.H.Manssour, S.S. Furuie, S.D. Olabarriaga, C.M.D.S. Freitas. Visualizing Inner Structures in Multimodal Volume Data. *Proceedings of SIBGRAPI - Brazilian Symposium on Computer Graphics and Image Processing*. Outubro, 2002. IEEE Computer Society. Fortaleza, Brazil.

[7] ITK Home Page, Kitware Inc. *National Library of Medicine Insight Segmentation and Registration Toolkit*. Disponível em http://www.itk.org/. Acesso em maio de 2005.

[8] J. Nielsen, J. *Usability Engineering*. London: AP Professional, 1993.

[9] J. Rubin. *Handbook of Usability Testing: How to Plan, Design e Conduct Effective Tests*. New York: John Wiley&Sons, 1994.

[10] B. Shneiderman. *Designing the User Interface: strategies for effective human-computer interaction*. Reading: Addison-Wesley, 1998.

[11] M. Winckler. Avaliação de Usabilidade de Sites Web IHC'2001. *Tutorial do Workshop sobre Fatores Humanos e Sistemas Computacionais*. Setembro, 2001. Florianópolis, SC, Brasil.

[12] L. Borges, L. Iglezias. *Ferramentas de Corte para Exploração Interativa de Imagens Médicas*. Trabalho de Conclusão de Curso, PUCRS, Brasil, 2004.

[13] Kitware Inc. *VolView Home Page*. Disponível em http://www.kitware.com/products/volview.html. Acesso em maio de 2005.

[14] Digital Imaging Unit. *Osiris Imaging Software*. Disponível em http://www.sim.hcuge.ch/osiris/01_Osiris_Presentation_EN.htm. Acesso em maio de 2005.

[15] A. Kaufman, R. Avila, L. Sobierajski, D. Bartz, T. He, L. Hong, H. Pfister. *VolVis - Volume Visualization System*. Disponível em http://www.cs.sunysb.edu/ vislab/volvis_home.html. Acesso em maio de 2005.

[16] National Institutes of Health. *ImageJ Home Page*. Disponível em http://rsb.info.nih.gov/ij. Acesso em maio de 2005.

[17] National Electrical Manufacturers Association. *DICOM Home Page*. Disponível em http://medical.nema.org. Acesso em maio de 2005.

[18] A. Glassner. *Graphics Gems*. (The Graphics Gems Series). AP Professional, 1990.

[19] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. *A System of Patterns: Pattern-Oriented Software Architecture*. West Sussex, UK: John Wiley & Sons, 1997.

[20] C. Larman. *Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design*. Upper Saddle River, NJ: Prentice-Hall, 1997.

[21] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, 1995.

[22] S. Barré. *dicom2 Home Page*. Disponível em http://www.barre.nom.fr/medical/dicom2. Acesso em maio de 2005.

[23] Bill Spitzak et al. *Fast Light Toolkit (FLTK) Home Page*. Disponível em http://www.fltk.org/. Acesso em maio de 2005.

Ulisses: Um Navegador Conceptual para *Topic Maps*

Giovani Rubert Librelotto*
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
grl@di.uminho.pt

and

José Carlos Ramalho
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
jcr@di.uminho.pt

and

Pedro Rangel Henriques
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
prh@di.uminho.pt

Resumo

A norma ISO-IEC 13250 – *Topic Maps* – faz a ponte entre os domínios de representação de conhecimento e gestão de informação. Tópicos e associações formam uma rede semântica estruturada sobre os recursos de informação. Este artigo tem por objectivo a visualização eficiente desta camada semântica, a qual é um assunto crítico porque topic maps podem conter milhões de tópicos e associações. Este artigo é dividido em 3 partes. Primeiro, apresenta-se brevemente os conceitos de *Topic Maps*. Então, são revistas algumas técnicas de visualização de *Topic Maps*. Finalmente, é descrita a ferramenta de visualização desenvolvida e descrita a forma de ser utilizada – e aprimorada – para a visualização de *Topic Maps*.

Palavras chaves: Semantic Web, Topic Maps, Visualização de Topic Maps, Ontologia, Grafos.

Abstract

The ISO standard ISO-IEC 13250 – *Topic Maps* – provide a bridge between the domains of knowledge representation and information management. Topics and topic associations build a structured semantic link network above information resources. This research aims at visualizing this semantic layer efficiently, which is a critical issue as topic maps may contain millions of topics and associations. This paper has 3 parts. First, we depict briefly basic *Topic Maps* concepts. Then, we review a few topic map visualization techniques. Finally, we describe the visualization tool we developed and study how this tool may be used – and enhanced – for *Topic Maps* visualization.

Keywords: Semantic Web, Topic Maps, Topic Maps Visualization, Ontology, Graphs.

*Bolsista CNPq - Brasil

1 Introdução

Topic Maps [1] é um formalismo para representar conhecimento acerca da estrutura de um conjunto de recursos de informação e para o organizar em tópicos. Esses tópicos possuem ocorrências e associações que representam e definem relacionamentos entre os tópicos. A informação sobre os tópicos pode ser inferida ao examinar as associações e ocorrências ligadas ao tópico. Uma colecção desses tópicos e associações é designada *topic map*. *Topic Maps* pode ser visto como um paradigma que permite organizar, manter e navegar através da informação, permitindo transformá-la em conhecimento.
Tendo em vista que os *Topic Maps* são grafos compostos por tópicos (onde cada tópico representa um tema, identifica recursos e está associado com outros tópicos), o *Ulisses* foi imaginado e desenvolvido para providenciar navegadores estáticos que permitem percorrer estas redes de conceitos. Esses navegadores são compostos de páginas HTML que descrevem os tópicos e usam a ideia de hiper-ligações para implementar as associações e as ligações às ocorrências.
A ideia de navegação conceptual reflecte a forma que a mente humana pensa: baseado em informações associadas. Por exemplo, quando se pensa no *Brasil*, automaticamente pode-se pensar:

- é um país que está situado na América do Sul (associação entre *país* e *continente*, no contexto *geografia*);
- possui mais de 170 milhões de habitantes (ocorrência do tipo *população*);
- é governado pelo presidente *Luís Inácio Lula da Silva* (associação entre *país* e *presidente*, no contexto *política*);
- é o actual campeão do mundo de futebol (associação entre *país* e *futebol*, no contexto *desporto*).

Assim, o que se pretende neste artigo é apresentar essa ferramenta para visualização de *Topic Maps* baseada em conceitos de navegação em grafos, chamada *Ulisses*. No contexto do *Metamorphosis*, o *Ulisses* possui a finalidade de criar interfaces web a partir dos topic maps gerados e validados pelos outros módulos desta arquitectura, respectivamente *Oveia* [10] e *XTche* (conforme apresentado na Figura 2).
Como se percebe do que foi dito acima, o *Ulisses* não se limita ao contexto do *Metamorphosis*; isto significa que ele possui a capacidade de gerar interfaces web para todo e qualquer topic map, que segue a sintaxe *XML Topic Maps* (XTM) ou que esteja armazenado no modelo relacional *BD Ontologia* [10].
Para ilustrar as ideias aqui introduzidas, esse artigo inicia apresentando ontologias e a norma *Topic Maps* na Secção 2. A Secção 3 apresenta o *Metamorphosis*, projecto no qual se encaixa o *Ulisses*. Uma técnica de visualização de *Topic Maps* encontra-se na Secção 4. O *Ulisses* é apresentado na Secção 5. A Secção 6 discute os trabalhos relacionados. Por fim, uma síntese do artigo e os trabalhos futuros são apresentados na conclusão.

2 Ontologias e Topic Maps

Uma ontologia é uma especificação ou *formalização de uma conceptualização* [6]. Alternativamente, uma ontologia pode ser entendida como uma teoria lógica a qual dá uma explicação explícita de uma conceptualização, projectada para ser compartilhada por agentes para vários objectivos [7]. Uma conceptualização é um conjunto de conceitos e suas relações entre si.
Na área de Sistemas de Informação, na qual se encaixa este trabalho, ontologia é definida como um conjunto de conceitos e termos ligados entre si (numa rede) que podem ser usados para descrever alguma área do conhecimento ou construir uma representação para o conhecimento [15]. Segundo Chandrasekaran [2], ontologias são teorias de conteúdo sobre os tipos de objectos, propriedades de objectos e relacionamentos entre objectos que são possíveis em um domínio de conhecimento específico.
Historicamente, a norma *Topic Maps* (ISO 13250) [1] foi definido para facilitar a fusão de diferentes esquemas de índices. Um formato comum para a anotação, usado para a indexação, é um passo crucial em direcção ao objectivo da interoperabilidade entre esquemas de índices. O que é necessário ainda é a interoperabilidade semântica. Enquanto que a especificação *Topic Maps* garante interoperabilidade sintáctica, ontologias provém interoperabilidade semântica. Se for construído a partir de uma ontologia válida, *Topic Maps* podem prover interoperabilidade semântica não somente entre cada topic map, mas entre as aplicações que usam-nas.
Topic Maps é um formalismo para representar conhecimento acerca da estrutura de um conjunto de recursos de informação, organizando-o em *tópicos*. Esses tópicos têm ocorrências em recursos de informação e associações que representam e definem os relacionamentos entre os tópicos. A informação sobre os tópicos pode ser inferida ao examinar as associações e ocorrências ligadas ao tópico. Uma colecção desses tópicos e associações é chamada de topic maps, conforme apresentada na figura 1.
Os *Topic Maps* podem ser definidos como uma descrição de um ponto de vista sobre uma colecção de recursos, organizado formalmente por tópicos, e pela ligação de partes relevantes do conjunto de informação aos tópicos apropriados [14]. Um mapa de tópicos expressa a opinião de alguém sobre o que os tópicos são, e quais as partes do conjunto de informação que são relevantes para cada tópico.

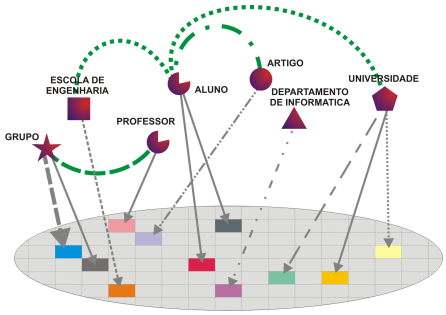


Figure 1: *Topic Maps*

Permitindo criar um mapa virtual da informação, os recursos de informação mantém-se em sua forma original e não são modificados. Então, o mesmo recurso de informação pode ser usado de diferentes formas, por diferentes mapas de tópicos. Como é possível e fácil modificar um mapa, a reutilização da informação é conquistada.

3 *Metamorphosis*: Um toolkit para *Topic Maps*

A motivação para o desenvolvimento do *Metamorphosis* [8] veio de algumas situações que surgiram no contexto de alguns projectos de desenvolvimento de software:

- Para conseguir interoperabilidade semântica em sistemas heterogéneos de informação, pois o significado da informação que é intercambiada deve ser compreendido por todos os sistemas [11]. Conflitos semânticos ocorrem toda vez que dois contextos não usam a mesma interpretação da informação, ou seja, quando há uma ambiguidade. Contudo, o uso de ontologias para a explicação do conhecimento implícito e oculto é uma possível abordagem para resolver o problema da heterogeneidade semântica.
- Muitas vezes, para se testar algumas funcionalidades de um sistema que se está a desenvolver é necessário criar uma interface Web para realizar esses testes. Normalmente, estas interfaces não têm grandes requisitos quanto à aparência, o que se pretende é que sejam desenvolvidas o mais rapidamente possível.
- Quer-se expor na Web um sistema de informação. Este sistema é composto por bases de dados, documentos XML ou documentos PDF. Quer-se que todos os items de informação estejam acessíveis via Web e para isso constroem-se os primeiros índices. Estes índices acabam por ser enormes excedendo a capacidade dos browsers da Internet. Podia pensar-se em fraccioná-los alfabeticamente, mas há situações em que isso não é possível nem recomendável. Mas, é sempre possível arranjar um método para fraccionar a informação conceptualmente. É aqui que se começa a discutir a organização dos recursos de informação e onde se encaixa o *Metamorphosis*. Não é necessário alterar nenhum dos recursos de informação, apenas é necessário criar uma ontologia para esses recursos que reflicta a visão que se quer para eles.

A Figura 2 dá uma visão geral da arquitectura do *Metamorphosis*. Esta arquitectura pode ser descrita da seguinte maneira:

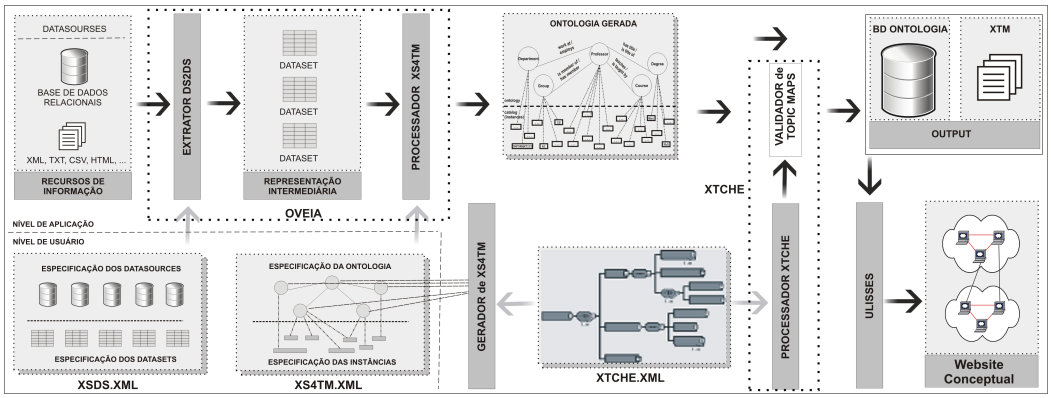


Figure 2: Arquitectura do *Metamorphosis*

(1) Camada de recursos de informação: Este componente é composto pelos vários recursos de informação: documentos XML, páginas Web, bases de dados, ... O *Metamorphosis* não interfere com nenhum deles, apenas utiliza parte da informação de cada um para construir a ontologia ou rede semântica.

(2) Especificações XSDS e XS4TM: São documentos XML que fornecem informações precisas sobre onde aceder para extrair as partes da informação necessárias para a construção da ontologia; este componente é descrito em detalhes em [10].

(3) *Oveia*: Este componente utiliza a especificação XS4TM para ir aos recursos de informação buscar a informação de que necessita para construir a ontologia. O *TM-Builder* [9] é um extractor de ontologia em documentos XML. O *Oveia* [10] é um extractor de topic maps em sistemas heterogéneos de informação, o qual foi o sucessor do *TM-Builder*, suprindo várias das carências da versão anterior.

(4) TM gerado: Esta é a especificação do topic map de acordo com a sintaxe XTM. Além da possibilidade de armazenamento dos topic maps gerados em formato XTM, os topic maps gerados pelo *Oveia* também podem ser armazenados em uma representação relacional. Para isso, foi mapeada a norma XTM para um modelo relacional; este modelo é chamado de *BD Ontologia*.

(5) Especificação *XTche*: É uma linguagem de especificação de restrições para topic maps, baseada nos requisitos de TMCL (*Topic Map Constraint Language*) [12] definidos pela ISO/IEC. Permite especificar regras para a validação de topic maps.

(6) Processador *XTche*: Este componente é responsável pela validação de topic maps de acordo com um conjunto de restrições especificado em *XTche*. Se os topic maps a serem validados estiverem de acordo com a especificação *XTche*, nada acontece; caso apresentem alguma irregularidade, mensagens de erros serão mostradas, indicando os pontos em que o topic map não está correto.

(7) *Ulisses*: Este é o componente que está a ser discutido com detalhe neste artigo. O *Ulisses* toma como entrada um topic map e produz uma visualização na web, de acordo com algumas regras. O *Ulisses* tanto fornece a navegação conceptual a partir da sintaxe XTM, como a partir do modelo relacional apresentado no *Oveia*.

(8) Website: Este é website final através do qual é possível navegar por entre os vários recursos de informação que compõem o sistema de informação original.

A próxima secção abordará a visualização de Topic Maps, que é o objectivo do Ulisses – um gerador de navegação conceptual para a norma ISO 13250 Topic Maps – que será apresentado em detalhes na Secção 5.

4 Visualização de *Topic Maps*

Antes de apresentar o *Ulisses*, esta secção introduz um modo de visualização de *Topic Maps*, que é o método seguido pelo próprio navegador.

4.1 Visualização de Tópicos e Associações

Como dito anteriormente, um topic map pode ser visto como um grafo, onde os tópicos podem ser vistos como os nodos e os relacionamentos como os arcos. Cada tópico é representado como um nodo no grafo. Assim, a visualização de cada nodo – disponibilizada quando o navegador se posiciona sobre um tópico em específico – deve conter todas as suas características: seus tipos, suas instâncias, seu indicador de tema, seus nomes e suas ocorrências, além de incluir uma referência a todas as associações a qual ele faz parte.
Os nodos são rotulados usando os nomes dos tópicos. Desta forma, sempre que um nodo está relacionado com outros (sejam relações classe/instância ou associações propriamente ditas), o rótulo do nodo que está relacionado é o seu próprio nome.
Cada associação binária pode ser representada como um arco conectando os dois tópicos. Ao se posicionar em determinada associação do grafo, as informações sobre a mesma serão apresentadas. Por exemplo: em uma associação do tipo *orientar*, os tópicos participantes são visualizados (*Giovani* e *Pedro Henriques*) e os respectivos papéis de actuação (*orientando* e *orientador*), além do contexto onde ela está inserida (*ensino*). Por sua vez, os papéis de actuação são usados para designar os arcos que saem desse nodo (isto é, para descrever as associações nas quais o tópico actua).

4.2 Filtragem em *Topic Maps*

Um topic map pode conter milhões de tópicos e associações. Portanto é essencial seleccionar a informação relevante pois pode ser impossível mostrar todo o conjunto de tópicos e associações de uma forma eficiente.
Técnicas de filtragem de informação são necessárias para seleccionar e apresentar somente a informação relevante. A ferramenta aqui apresentada habilita os utilizadores a filtrar tópicos e associações de acordo com seu tipo ou pelos papeis de actuação em associação, por exemplo.
É possível fazer aqui uma analogia entre *Topic Maps* e mapas geográficos. Geralmente, não se encontra toda a informação desejada sobre um país em um único mapa; em vez disso, existem sub-mapas específicos de diversos contextos, tais como: mapas topográficos, mapas políticos, mapas económicos, etc.
Da mesma forma, tópicos e associações podem ser classificados em diferentes contextos. Sendo assim, sub-mapas[1] distintos dentro de um topic map poderão fornecer as informações necessárias ao utilizador de acordo com seu interesse. Se ele estiver interessado em *teatros*, os tópicos relevantes seriam *autor*, *comédia*, *cultura*, *actor*, etc. Esta é uma forma de filtrar a informação de acordo com um contexto específico.
Uma vez filtrados os tópicos e associações, eles necessitam ser representados eficientemente. Uma sugestão para a redução do número de tipos a serem representados é a agregação de tópicos e associações com um algoritmo de classificação, como o *galois lattices* [5]; este algoritmo diz que se pode agrupar objectos que compartilhem propriedades comuns. Esses grupos são chamados de classes. Portanto, pode-se somente distinguir classes de tópicos na representação em vez de diferenciar todos os tópicos. Por exemplo, os tópicos *Brasil*, *América do Sul* e *Brasília* podem ser representados da mesma maneira porque eles pertencem à mesma classe *lugar*. Este mecanismo de classificação torna possível visualizar *Topic Maps* com diferentes níveis de detalhes.
Obviamente, a informação visualizada é menos precisa, mas isso pode ser aceitável dependendo do propósito da navegação. Contudo, é possível visualizar uma informação mais precisa quando o utilizador foca em uma parte específica do topic map. Por exemplo, ao estar situado no tópico *Brasília*, a informação mostrada será que *Brasília é uma cidade*, o que é mais preciso que simplesmente afirmar que *Brasília é um local*. O mesmo acontece quando verifica-se a associação entre *Brasília* e *Brasil*; a informação apresentada é *Brasília é uma cidade situada no Brasil*.

4.3 Abordagem escolhida na criação do *Ulisses*

A fim de evitar a geração de um grafo para a visualização do topic map, o desenvolvimento do *Ulisses* seguiu uma abordagem paralela: a criação de um conjunto de páginas HTML, onde cada página representa um tópico ou uma associação contido no topic map.
Os *links* determinam as relações entre os nodos; desta forma, os *links* são identificados pelos nomes dos tópicos que eles representam. Por exemplo, na página HTML gerada para o tópico *Brasília* (conforme citado na subsecção anterior), haveria a menção de uma associação com o tópico *Brasil*, onde a frase "*é uma cidade que está situada no*" representa o papel de actuação de *Brasília* perante *Brasil*. Desta forma, a associação seria apresentada da seguinte forma:

- *Brasília* é uma cidade que está situada no *Brasil*

A palavra *Brasil*, nesta frase, possui uma ligação à página HTML que representa o tópico *Brasil*.
O algoritmo de funcionamento seguido pelo *Ulisses* para a criação das páginas HTML, as quais oferecerão a navegação conceptual a partir do topic map, pode ser descrito assim:

1. Definição da hierarquia de tópicos do topic map, percorrendo-se todos os tópicos e ligando-os de acordo com as relações classe/instância. Se no final desse processo verificar que não há um tópico raiz, é então criado um novo tópico que servirá de raiz para o topic map. Este novo tópico terá, como suas instâncias, todos os tópicos que não possuem tipos. Desta forma, todas as sub-árvores estarão conectadas entre si;
2. Criação de uma página inicial para o topic map, usando o nome do tópico raiz para a sua identificação e mostrando a hierarquia de tópicos no lado esquerdo da página;
3. Criação de uma página que oferece 4 formatos distintos de visualização da ontologia do topic map (tópicos abstractos), conforme será descrito na Subsecção 5.1.2;
4. Criação de uma página com o índice alfabético contendo todos os tópicos;
5. Criação de uma página para cada tópico, onde é apresentado os seus tipos, seu indicador de tema, suas ocorrências, as associações a qual ele faz parte e os tópicos que são suas instâncias. A página de cada tópico será armazenada fisicamente com nome formado por seu identificador com a extensão "html". Deste modo, quando se faz uma referência a algum tópico, cria-se um *link* com o identificador do tópico relacionado;

[1] Um sub-mapa é o conjunto de tópicos e associações que representam um sub-domínio, dentre vários que podem existir num topic map. Um mesmo tópico pode estar inserido dentro de vários sub-mapas, de acordo com as relações que ele possui.

6. Criação de uma página para cada associação, indicando seu tipo e seu contexto, além de seus membros e respectivos papeis de associação, onde os *links* seguem a estratégia acima: são criados a partir do identificador de cada tópico.

Seguindo esses passos, o website que corresponde a navegação conceptual do topic map será criado e armazenado numa directoria do sistema operativo.

5 *Ulisses*– o navegador

A ideia sobre a qual se baseou o *Ulisses* é a ideia da navegação conceptual, a qual pode ser descrita como: quando se está posicionado sobre um certo conceito, a ferramenta de navegação mostrará as informações associadas a este conceito em particular; se for escolhido algum dos outros conceitos relacionados, a navegação muda para a visão deste novo conceito; se for escolhido algum dos recursos de informação, o sistema mostrará o conteúdo do próprio recurso. O *Ulisses* foi projectado com base em três razões principais:

- Ser um protótipo eficiente, diminuindo o tempo de criação de um navegador para *Topic Maps*;
- Criar uma ferramenta que usa *Topic Maps* que seja facilmente distribuída, instalada e usada por todos;
- Ser baseado em XML.

Em termos de implementação, foi estabelecido que a navegação seria realizada através de *web browsers*. Então, todas a informação sobre cada conceito seria apresentada em páginas Web, não necessitando assim de utilitários extras, pois todo o computador adaptado à internet teria capacidade de navegar no topic map, independente de plataforma ou sistema operativo.
Quando o topic map estiver representado em XTM, a implementação desse componente de navegação é reduzida a transformações de XML para HTML. O *Ulisses* oferece, então, duas possibilidades: transformação do XTM com XSL (website estático) – Subsecção 5.1; e transformação do XTM em tempo de execução (website dinâmico) – Subsecção 5.2.
Além destas duas alternativas para a navegação, o *Ulisses* também é capaz de gerar um navegador para *Topic Maps* representados em base de dados relacionais – Subsecção 5.3. Este navegador surgiu de uma necessidade originada no âmbito do *Metamorphosis*, pois os topic maps gerados pelo *Oveia* podem ser armazenados tanto na sintaxe XTM, como na *BD Ontologia* [10].
Recorda-se, a propósito, que embora o *Ulisses* tenha sido investigado no patamar filial do *Metamorphosis* para fornecer a navegação desejável sobre o topic map extraído, é um componente independente, usado além do *Metamorphosis*.
Visando tornar o *Ulisses* mais abrangente, foram adicionadas folhas de estilos XSL[2] que permitem transformar os topic maps representados na sintaxe HyTime para a sintaxe XTM [13], de modo a permitir gerar navegadores para topic maps em HyTime. Assim, a navegação conceptual pode ser gerada para os principais formatos XML para a representação de *Topic Maps*. Além disso, o tradutor das sintaxes AsTMa=[3] e LTM[4] para a sintaxe XTM pode ser utilizado para os topic maps que se encontram num destes formatos não-XML.

5.1 Navegação em *Topic Maps* baseada em transformações XSL

A variante mais utilizada do *Ulisses* é aquela que gera o website de uma só vez, isto é, transforma previamente (antes de se fazer a navegação) o topic map, em XTM, num conjunto de páginas HTML que não voltam a ser geradas em toda a navegação (site estático) [8]. Por isso, esta variante do *Ulisses* é desenvolvida em XSL, criando websites em HTML com algumas componentes em *Javascript*.
O algoritmo de funcionamento desta variante do *Ulisses* foi descrito na Secção 4.3.
Esta versão do *Ulisses* oferece opções que permitem rápidas modificações em todo o website, como por exemplo, alterações em termos de *layout* (cores e tamanho das fontes), comentários a serem incluídos nas páginas criadas e a inserção de imagens.
A seguir, cada formato de página gerado pelo *Ulisses* é apresentado, nomeadamente: a página inicial e a página que mostra a ontologia do topic map, a página que descreve um tópico e, por fim, o modelo de página para as associações.

5.1.1 Página principal do topic map

A página inicial (*home page*) da navegação num topic map no *Ulisses* é apresentada na Figura 3. Esta imagem expõe no seu lado esquerdo, a hierarquia de classes obtida a partir da ontologia, na forma de um menu que permite aceder qualquer tipo de tópico do topic map – item (1).

[2] `http://www.w3.org/Style/XSL/`
[3] XTM::AsTMa – `http://cpan.uwinnipeg.ca/htdocs/XTM/XTM/AsTMa.html`
[4] XTM::LTM – `http://cpan.uwinnipeg.ca/htdocs/XTM/XTM/LTM.html`

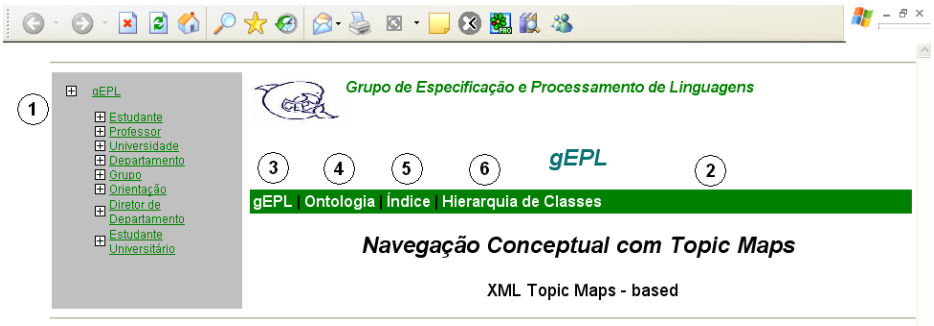


Figure 3: Visualização inicial de um topic map no *Ulisses*

No corpo do navegador, há uma barra de navegação – item (2) – que se encontra em todas as páginas geradas pelo *Ulisses*. Esta barra oferece 4 opções navegacionais:

Home: a primeira opção faz com que a navegação volte à página inicial do topic map, independente em que nodo a visualização do topic map se encontre num determinado instante. Essa opção é referenciada pelo nome do tópico raiz do topic map. Na Figura 3, *gEPL*[5] é o nome do tópico raiz, por isso aparece na barra de navegação em todas as páginas geradas para este topic map – item (3);

Ontologia: ligação que remete à página que descreve a hierarquia de classes da ontologia do topic map – item (4);

Índice: indica a página que apresenta um índice alfabético dos tópicos – item (5);

Hierarquia de Classes: propicia a visualização da hierarquia de classes contida na ontologia do topic map em um formato gráfico – item (6).

5.1.2 Menu Ontologia

A ontologia – item (3) na Figura 4 – pode ser descrita em 4 partes, de acordo com as opções do sub-menu mostrado na Figura 4 e descritos abaixo:

Temas: apresenta a árvore hierárquica completa formada pelos tipos de tópicos e suas instâncias – item (1);

Relacionamentos: mostra todas as associações encontradas no topic map – item (2);

Papeis de Actuação: descreve os papeis de actuação e as associações nas quais cada papel está envolvido – item (3);

Recursos de Informação: lista todas as ocorrências encontradas no topic map agrupadas por seus tipos – item (4).

5.1.3 Visualização de um tópico

A visualização de um tópico pode ser composta por 6 partes, conforme pode ser visto na Figura 5, onde se apresenta a visualização do tópico *Giovani R Librelotto*:

Nome do topic map e barra de navegação: no cabeçalho do topic map aparece o nome do tópico raiz (neste caso, *Giovani R Librelotto*). Abaixo, encontra-se a barra de navegação, com as opções de voltar à página principal, aceder a ontologia, o índice geral e a hierarquia de classes do topic map – item (1);

Tipo do tópico: é uma ligação ao tipo do tópico em questão. No caso, o tópico *Giovani R Librelotto* é uma instância de *Estudante* – item (2);

Nome do Tópico: em destaque, o nome do tópico caracteriza o tópico em questão – item (3);

Recursos de Informação: as ocorrências do tópico, juntamente com seus tipos, são apresentadas nesta parte. Por exemplo: uma URL é definida como uma referência a um recurso do tipo *website* que caracteriza o tópico *Giovani R Librelotto* – item (4);

[5] *gEPL* refere-se ao Grupo de Especificação e Processamento de Linguagens da Universidade do Minho. Ver: `http://wiki.di.uminho.pt/wiki/bin/view/EPL/WebHome`

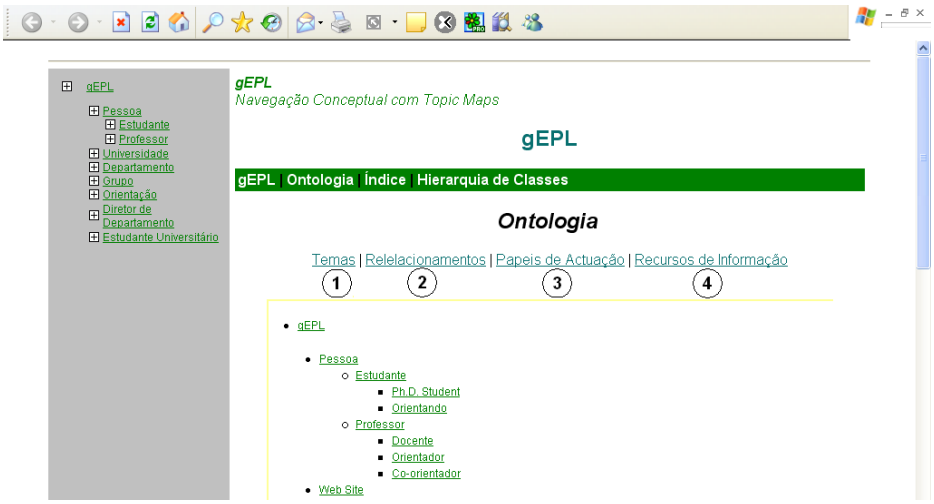


Figure 4: Visualização da ontologia de um topic map no *Ulisses*

Tópicos Relacionados: os tópicos que estão associados, assim como o papel de actuação do tópico em questão em cada associação, se encontram postos de forma que a leitura seja intuitiva. Por exemplo: *Giovani R Librelotto é orientado por Pedro R Henriques* define uma associação envolvendo dois tópicos (*Giovani R Librelotto* e *Pedro R Henriques*), além do papel de actuação do primeiro em relação ao segundo (*é orientado por*) – item (5);

Instâncias: lista dos tópicos que são instâncias do tópico em questão. Por exemplo: a visão do tópico *Estudante* apresenta o tópico *Giovani R Librelotto* e suas demais instâncias. Na Figura 5 não se encontra nenhuma instância, pois o tópico *Giovani R Librelotto* é um nodo folha na hierarquia de classes do topic map.

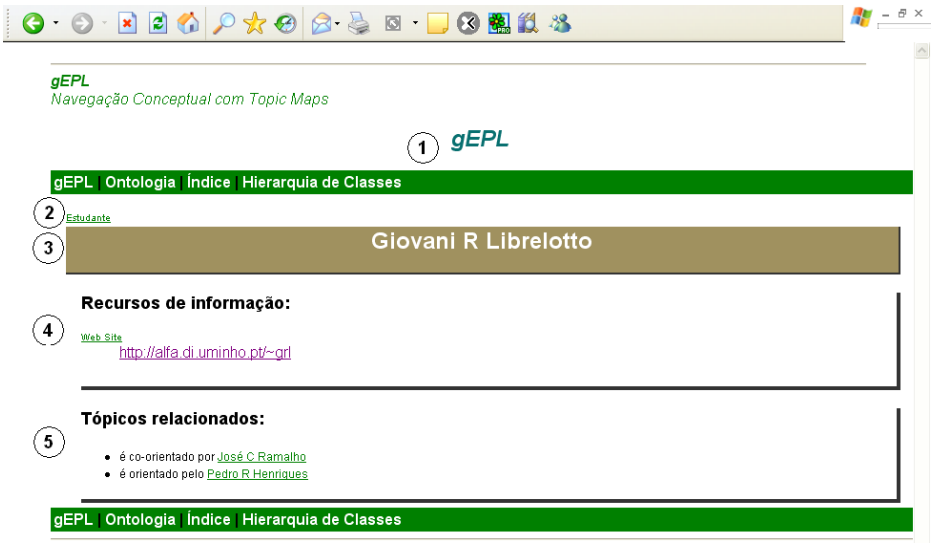


Figure 5: Visualização de um tópico no *Ulisses*

Dessa forma, todas as características de tópicos discutidas na Subsecção 4.1 encontram-se descritas numa página que segue o formato acima descrito.

5.1.4 Visualização de uma associação

A visualização de uma associação pode ser composta por 5 partes, conforme pode ser visto na Figura 6, onde se apresenta a visualização da associação *Chefe de Grupo*:

Nome do topic map e barra de navegação: assim como acontece na visualização de tópicos, o nome do tópico

aparece no alto da página (neste caso, *Chefe de Grupo*); logo abaixo encontra-se a barra de navegação – item (1);

Tipo da associação: é uma ligação ao tipo da associação. No caso, a associação *Chefe de Grupo* é uma instância de *gEPL*, o qual é o tópico raiz do topic map. Esse facto indica que a associação não possuia nenhum tipo e foi relacionado, pelo *Ulisses*, com o tópico raiz – item (2);

Nome da Associação: em destaque, o nome que caracteriza a associação em questão – item (3);

Tipos dos Tópicos Membros: em cada frase que determina a relação entre os tópicos membros, os seus tipos são citados; cada citação é um *link* para a página referente ao tipo de tópico – item (4);

Membros: os membros da associação constituem *links* para a páginas dos referidos tópicos. Por exemplo, o tópico *José C Ramalho* participa da associação *Chefe de Grupo* juntamente com o tópico *Grupo de Especificação e Processamento de Linguagens* – item (5);

Papeis de Actuação: cada membro da associação está relacionado aos demais membros através de um papel de actuação. Por exemplo, o tópico *José C Ramalho* possui o papel *é chefe de* na associação com o tópico o tópico *Grupo de Especificação e Processamento de Linguagens* – item (6).

Figure 6: Visualização de uma associação no *Ulisses*

Assim como ocorre com os tópicos, as associações também tem todas as suas características apresentadas em uma página que segue o formato acima descrito.

5.2 Navegação em *Topic Maps* com transformação em tempo de execução

As transformações em tempo de execução são implementadas com transformações XML, auxiliadas por uma CGI (*Common Gateway Interface*). A página inicial é uma chamada a uma CGI parametrizada à raiz do topic map (que vai servir de ponto de entrada para a navegação). A CGI aplica a transformação no topic map e produz uma visão HTML. Nessa visão, todos os *links* gerados são chamadas à mesma CGI mas com diferentes parâmetros, que indicam a localização da nova posição, ou seja, permitirá a visão do tópico relacionado.
Esta é a solução mais prática porque somente necessita de dois ficheiros: o topic map (especificado de acordo com a sintaxe XTM) e a CGI. Contudo, as transformações em tempo de execução consomem muito tempo, pois o topic map é interpretado em tempo de execução. Desta forma, a cada nova chamada a CGI, o topic map deve ser interpretado mais uma vez, para formar a visualização a ser disponibilizada no navegador. Por isto, esta solução tem sido utilizada apenas com topic maps considerados pequenos, ou seja, com um número inferior a 200 tópicos e associações.
A navegação no *Ulisses* baseado em CGI segue o modo descrito na Subsecção 5.1. Apesar da navegação ser gerada de uma forma distinta – em vez de uma página HTML para cada tópico e associação, usa-se uma chamada à CGI com o identificador do tópico ou associação desejado – a visualização do topic map é a mesma. Assim, as imagens apresentadas na subsecção anterior servem também para descrever essa variante do *Ulisses*.
Assim como a variante do *Ulisses* apresentada na subsecção anterior, esta variante baseada em CGI também permite a navegação sobre topic maps representados de acordo com a sintaxe XTM. Ou seja, se o topic map estiver em algum

outro formato, deve ser convertido para XTM (antes da geração da navegação) pelos processadores de HyTime[6], AsTMa= e LTM.

5.3 Navegação em *Topic Maps* armazenados na *BD Ontologia*

Conforme apresentado em [10], além da geração de topic maps em ficheiros XTM, o *Oveia* propicia também o armazenamento dos mesmos num formato relacional. Desta forma, em vez de ficheiros texto, uma base de dados, chamada *BD Ontologia*, funciona como um repositório de topic maps. A partir dessa base de dados, é possível navegar nos topic maps utilizando consultas SQL.
As consultas SQL podem ser realizadas para diferentes fins. Um exemplo pode ser visto no código abaixo, o qual apresenta um comando SQL para retornar as instâncias de um tópico, que se encontra na base de dados.

```
SELECT bn.baseNameString, t.* FROM topicRef tr
    inner join instanceOf [io] on tr.topicRefID = [io].topicRefID
    inner join instanceOfTopic iot on [io].instanceOfID = iot.instanceOfID
    inner join topic t on t.topicID = iot.topicID
    inner join baseName bn on t.topicID = bn.topicID
    WHERE tr.topicID = 1 AND bn.scopeID is null
ORDER BY bn.baseNameString
```

A consulta SQL acima busca, na *BD Ontologia*, todos os tópicos que são instâncias de um determinado tópico, ordenando-os pelo nome. Assim, entende-se que há uma consulta SQL implementada para cada tipo de interrogação que se pode vir a fazer no topic map, seja ela sobre tópicos, associações ou ocorrências.
As principais vantagens deste formato de navegação são:

- manter o topic map armazenado num modelo relacional. Pode-se, assim, aproveitar os recursos que uma base de dados relacional oferece para a gestão do topic map;
- realizar a navegação sobre o topic map actual; isso significa que se houver mudanças no topic map armazenado, elas automaticamente serão reflectidas na visualização;
- alternar o topic map a ser visualizado, sem necessitar abrir um novo navegador. Como uma única *BD Ontologia* pode armazenar diversos topic maps, é possível alterar o topic map em que se está a navegar, de uma maneira simples.

Contudo, essa variante do *Ulisses* também possui algumas desvantagens, tais como:

- necessita de um gestor de bases de dados (onde está o topic map a ser navegado) e de um servidor Web (por exemplo, o *TomCat*[7]) para o seu completo funcionamento;
- a velocidade da navegação depende de factores externos, tais como a base de dados e o servidor Web, pois as páginas são montadas em tempo de navegação.

6 Trabalhos Relacionados com o *Ulisses*

Actualmente, os navegadores (*browsers*) para *Topic Maps* mais conhecidos pela comunidade académica de Semantic Web são: *Omnigator*, *Topic Map Designer*, *xSiteable* e *TMNav*.
Actualmente, o *Ontopia Omnigator*[8] talvez seja o navegador de *Topic Maps* mais difundido. O *Omnigator* é uma aplicação que permite carregar e navegar sobre qualquer topic map, usando um browser para a Web. O objectivo do desenvolvimento do *Omnigator* foi incentivar o uso de *Topic Maps*, ensinando os princípios básicos do paradigma.
O *Omnigator* faz parte do *Ontopia Knowledge Suite* (*OKS*). Isto implica que algumas funcionalidades do *Omnigator* foram projectadas para trabalhar em conjunto com outros módulos do OKS. Contudo, o *Omnigator* é o único módulo integrante do *OKS* fornecido livremente; os demais módulos devem ser adquiridos. Outro inconveniente do *Omnigator* é que a versão disponibilizada livremente está limitada a 5000 tópicos e associações; desta forma, quando um topic map possuir uma quantidade superior a este limite, a navegação não é permitida ao utilizador.
Apesar deste inconveniente comercial e da limitação imposta na versão distribuída livremente, o *Omnigator* é considerado uma ferramenta completa e eficiente, que produz um navegador com uma interface muito agradável e simples de manusear. Contudo, o *Omnigator* é um interpretador – processa o topic map quando carrega a especificação XTM indicada – o que requer a sua presença na máquina do utilizador final, ou então o recurso a um servidor Web que terá de estar activo numa determinada máquina.

[6] XTM::HyTyme – `http://www.cogx.com/xslt4tm2xtm.html`
[7] `http://jakarta.apache.org/tomcat/`
[8] Disponível em: `http://www.ontopia.net/omnigator/models/index.jsp`

Constata-se, na prática, que o *Omnigator* é uma óptima solução para a fase de desenvolvimento, quando o topic map está constantemente a ser alterado[9]. Nessa fase, a navegação proporcionada pelo *Omnigator* pode ser utilizada para certas verificações, como por exemplo, a correcção dos nomes dos tópicos e dos seus contextos. Porém, numa situação de produção, em que o topic map está estável, é preferível usar uma solução em que o visualizador possa ser aberto por qualquer *browser* sem requerer a presença do processador.
Foi precisamente este requisito que nos levou a desenvolver o *Ulisses* que, sendo comparável a um compilador, gera um conjunto de páginas HTML estáticas. Assim, o *Ulisses* é invocado uma vez e o resultado permite efectuar navegações com qualquer *browser* que o utilizador final escolha.
Entretanto, uma vantagem significativa do *Omnigator* é o seu módulo chamado *Vizigator* [4]. O *Vizigator* propicia uma navegação gráfica de *Topic Maps* fornecendo: uma visão geral da estrutura da informação, de forma intuitiva e instantânea; uma travessia no grafo formado pelo topic map; a habilidade de ver e entender os relacionamentos; visões em diferentes níveis de granularidade; e acesso intuitivo a modelos de dados não-familiares ao utilizador.
A visualização gráfica fornecida pelo *Vizigator* se caracteriza, portanto, num diferencial a favor do *Omnigator*, por propiciar duas navegações distintas sobre *Topic Maps*: numa visão HTML contendo toda a informação sobre cada tópico (como faz o *Ulisses*) ou numa visão gráfica, apresentando o grafo propriamente dito (como faz o *Topic Map Designer*).
O *Topic Map Designer*[10] é um aplicativo que permite a edição de *Topic Maps* e sua consequente visualização gráfica. Contudo, ele não tem como objectivo ser um ambiente completo para a criação de *Topic Maps* e suporta somente topic maps no formato HyTime e possui limitações quanto ao tamanho de topic maps (não podem ter mais de 100 tópicos).
O *xSiteable*[11] é uma ferramenta de desenvolvimento de websites criada em XSLT, com um pacote de administração PHP. De modo geral, possui características muito similares ao *Ulisses*, quanto à formação do website, representação das características dos tópicos e associações, assim como na visualização de todas estas informações em uma página HTML. Ambas utilizam tecnologia XSLT para a geração de páginas HTML, a partir de um documento XTM.
Porém, o *xSiteable* apenas possui a opção de criar uma página HTML por tópico. Ao contrário do *Ulisses*, o *xSiteable* não permite a criação de um website dinamicamente, (conforme apresentado na Subsecção 5.2). Além disso, não suporta qualquer outro formato de topic maps, além de XTM e CSXTM[12]. O *Ulisses*, por sua vez, permite a navegação em topic maps armazenados na *BD Ontologia*.
A mesma vantagem acima – permitir a navegação de topic maps armazenados num modelo relacional – o *Ulisses* possui sobre o *TMNav*. O *TMNav*[13] é uma aplicação *Java/Swing* para propiciar a navegação em topic maps que utiliza uma interface baseada em grafos. Ele funciona sobre o aplicativo *TM4J*, permitindo assim uma navegação em páginas Web ou num grafo dinâmico que utiliza a biblioteca *TouchGraph*[14]. Com isso, a desvantagem do *Ulisses* é justamente o facto de não fornecer uma visualização diagramática do grafo, como faz tanto o *TMNav*, como o *Omnigator*/*Vizigator*.

7 Conclusão

Em resumo, o *Ulisses* representa uma ferramenta com objectivos claros (navegação conceptual em *Topic Maps*), baseada em conceitos de navegação em grafos, a qual utiliza tecnologias bastante difundidas para sua implementação, não o tornando dependente de plataforma ou aplicativo específico.
Actualmente, o *Metamorphosis* tem sido utilizado em vários projectos de pequena e média dimensão. Até ao momento, provou ser uma boa ferramenta de prototipagem. Com isso, a necessidade de uma ferramenta que disponibilize uma navegação conceptual é importante e indispensável, pelo que se acrescentou o *Ulisses* ao *Metamorphosis*. Assim, as interfaces Web – que disponibilizam o conhecimento representado no topic map – são criadas rapidamente e sem grandes dificuldades por parte dos utilizadores.
O *Ulisses* fornece uma navegação completa sobre *Topic Maps*, gerados ou não a partir de outras ferramentas. A representação do conhecimento é apresentado de uma forma simples e precisa, formando uma rede semântica baseada em tópicos e associações. Quando se acede às informações referente a um determinado tópico, visualizam-se suas características (o seu tipo, suas instâncias, seus identificadores de tema, seus nomes e suas ocorrências) e as associações relacionadas com este tópico (incluindo os papéis de associação actuado por ele e os tópicos associados).
Apresentando três tipos distintos de navegadores, o *Ulisses* abrange praticamente todos os formatos de representação de *Topic Maps*. Cada navegador possui sua aplicação ideal: o navegador baseado em XSL é indicado para qualquer tamanho de topic maps, pois gera uma interface completa, independente da quantidade de tópicos e associações; o

[9] Como o *Omnigator* interpreta a especificação na hora em que o serviço é requisitado, qualquer alteração que tenha sido produzida no ficheiro XTM é logo reflectida no visualizador.
[10] Topic Map Designer: `http://www.topicmap-design.com/en/tutorial.htm`
[11] xSiteable: `http://www.shelter.nu/xsiteable/`
[12] CSXTM é um formato de representação de *Topic Maps* compacto, simplificado e baseado em XTM. Desenvolvido pelos mesmos criadores do *xSiteable*, utiliza-se folhas de estilos XSL para a conversão de XTM para CSXTM, e vice-versa. Detalhes em `http://www.shelter.nu/csxtm.html`
[13] TMNav: `http://tm4j.org/tmnav.html` – TM4J: `http://tm4j.org/`
[14] TouchGraph: `http://touchgraph.sourceforge.net/`

navegador baseado em CGI é indicado para uma rápida visualização de pequenos topic maps (com até 200 tópicos e associações); por fim, o navegador para base de dados relacional acrescenta a possibilidade de navegação em topic maps armazenados na *BD Ontologia*.
No que diz respeito a trabalhos futuros, está a ser estudado um componente que irá permitir ao utilizador especificar o aspecto visual do website gerado (por enquanto, apenas há uma configuração da interface). Desta forma, a visualização da ontologia representada no topic map visualizado pode ser apresentada da forma mais propícia.
Em termos de trabalho actual, outra família de navegadores está em desenvolvimento: navegadores gráficos SVG [3]. Estes navegadores se incorporarão ao *Metamorphosis*, tornando-o uma ferramenta cada vez mais completa, no que diz respeito a processadores para a norma *Topic Maps*.

References

[1] Michel Biezunsky, Martin Bryan, and Steve Newcomb. ISO/IEC 13250 - Topic Maps. ISO/IEC JTC 1/SC34, December 1999. `http://www.y12.doe.gov/sgml/sc34/document/0129.pdf`.

[2] B. Chandrasekaran. What Are Ontologies, and Why do We Need Them? In *IEEE Inteligent Systems and their applications*, volume vl 9, n 1. IEEE, January 1999.

[3] Jon Ferraiolo and Dean Jackson. Scalable Vector Graphics (SVG) 1.1 Specification. World Wide Web Consortium, January 2003. `http://www.w3.org/TR/SVG/`.

[4] Pamela Gennusa. Ontopia's Vizigator(tm) - Now you see it! In *XML 2004 Conference and Exposition*, Washington D.C., U.S.A, 2004. IDEAlliance. `http://www.idealliance.org/proceedings/xml04/papers/311/311.html`.

[5] R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on galois (concept) lattice. In *Computational Intelligence*, volume 11(2), 1995.

[6] Thomas R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[7] Nicola Guarino and P. Giaretta. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. Ed. Amsterdan: ISO Press, 1995.

[8] Giovani R. Librelotto, José C. Ramalho, and Pedro R. Henriques. Ontology driven Websites with Topic Maps. In *The International Conference on Web Engineering*, Oviedo, Spain, 2003.

[9] Giovani R. Librelotto, José C. Ramalho, and Pedro R. Henriques. TM-Builder: Um Construtor de Ontologias baseado em Topic Maps. In *XXIX Conferencia Latinoamericana de Informtica*, La Paz, Bolívia, 2003.

[10] Giovani Rubert Librelotto, José Carlos Ramalho, and Pedro Rangel Henriques. Extração de Topic Maps no Oveia: Especificação e Processamento. In *30ma Conferencia Latinoamericana de Informática (CLEI2004)*, 2004. ISBN 9972-9876-2-0.

[11] Steven R. Newcomb. A Semantic Integration Methodology. In *Extreme Markup Languages 2003: Proceedings*. IDEAlliance, 2003. `http://www.idealliance.org/papers/extreme03/html/2003/Newcomb01/EML2003%Newcomb01.html`.

[12] Mary Nishikawa and Graham Moore. Topic Map Constraint Language (TMCL) Requirements and Use Cases. ISO/IEC JTC 1/SC34 N0405rev, 2003. `http://www.isotopicmaps.org/tmcl/requirements.html`.

[13] Nikita Ogievetsky. XSLT stylesheets for converting ISO 13250 Topic Map documents into XTM 1.0 syntax. `http://www.cogx.com/xslt4tm2xtm.html`, 2000.

[14] Steve Pepper. The TAO of Topic Maps - finding the way in the age of infoglut. Ontopia, 2000. `http://www.ontopia.net/topicmaps/materials/tao.html`.

[15] W. Swatout and A. Tate. Ontologies. In *IEEE Inteligent Systems and their applications*, volume vl 14, n 1. IEEE, January 1999.

Catalogación y búsqueda semántica en un sitio web

Juan Barrios N. - Claudio Gutiérrez
Universidad de Chile, Departamento de Ciencias de la Computación,
Blanco Encalada 2120, Santiago, Chile
jbarrios@dcc.uchile.cl - cgutierr@dcc.uchile.cl

Resumen

La Web Semántica es una propuesta de la W3C que permite automatizar el procesamiento semántico de la información en la Web actual. Una de las aplicaciones que más se potencia con este enfoque es la *catalogación*, es decir, el proceso de creación de información agregada a nivel semántico. Este trabajo propone un modelo para la catalogación semi-automática de un sitio web a partir de la creación de un conjunto de metadatos sobre los contenidos de un sitio. A partir de esto crea un catálogo y ofrece al usuario distintos buscadores sobre estos conceptos semánticos. Este enfoque mejora los resultados de los buscadores sintácticos en el ámbito de intranets, donde las técnicas de recuperación de información clásica no han mostrado los éxitos que tienen en Internet global. Este artículo reporta el modelo, su implementación, un caso de estudio, y su comparación con buscadores sintácticos en un sitio.

Palabras claves: Catálogos, Metadatos, Web Semántica, Intranet, Búsqueda Semántica

Abstract

The Semantic Web is a proposal of the W3C to allow automatic processing of semantic information in the current Web. One of the applications which benefits more with this approach is *cataloguing*, that is, the process of creation of aggregate information at a semantic level. This work proposes a model for semi-automatic cataloguing of a Web site based on the creation of a set of metadata about the site contents. It builds a catalog and offers the user different search procedures based on semantic concepts. This approach improves results of syntactic search engines in the scope of intranets, where classical information retrieval techniques are far from having the success they enjoy in the global Internet. This paper reports a model, its design and implementation, tests it against a study case, and compares it with syntactic search engines in a single Web site.

Keywords: Catalog, Metadata, Semantic Web, Intranet, Semantic Search

1. Introducción

La Web Semántica es una extensión de la Web tradicional donde a la información publicada en lenguaje natural se le agrega un significado estructurado, con el objetivo de permitir que el contenido de un documento pueda ser procesado y entendido por una computadora [3]. Para aumentar la comprensión de los computadores, los humanos deben extraer la información relevante de cada documento y mantenerla como datos agregados o *metadatos*. Una de las aplicaciones más interesantes que ha potenciado este enfoque es la *catalogación*, debido a las técnicas proporcionadas para la creación de información agregada a nivel semántico. En particular, hoy tenemos las herramientas conceptuales para enfrentar la tarea de catalogación de páginas web. Los Directorios Web -como el *Open Directory Project*- son una demostración de las posibilidades que proporciona la catalogación dentro de la Web, como reunir sitios relacionados o realizar búsquedas restringidas a ámbitos temáticos.

La catalogación ha sido utilizada con anterioridad a la existencia de la Web, particularmente en las bibliotecas, donde se extrae información de cada libro por bibliotecarios especializados creando una ficha correspondiente bajo un formato y reglas definidas, formando un *catálogo*. Sin embargo, aún cuando este proceso sea una técnica antigua y esté estandarizada desde los años '60 en formatos como el MARC (Machine Readable Catalogue Format) [6], los documentos digitales de la Web no pueden catalogarse en el sentido

tradicional y estricto de una biblioteca [11]. Esto se debe a que existen características específicas de la información electrónica que hacen que un registro de metadatos de un documento electrónico (un texto, un sonido, una imagen digital, un programa, etc.) difiera de los registros catalográficos tradicionales de la información tangible (libros, revistas, etc.). Cualquier forma de catalogación debe tomar en cuenta, además, la naturaleza de la Web y sus principales dificultades para esta área: *Masividad*, *Dinamismo* y *Distribución* [2, 7].

Por otra parte, los buscadores sintácticos de Internet se han transformado en la principal puerta de acceso a la gran cantidad de información disponible en línea. Los usuarios incluso han comenzado a utilizarlos como "buscadores de respuestas", visualizando a la Web como un solo gran recurso que proporciona información sin importar de donde provenga ésta [14]. Diversos estudios de satisfacción demuestran la buena evaluación que tienen los buscadores de Internet por parte de los usuarios, por ejemplo, en un estudio de la *American Customer Satisfaction Index* publicado en agosto del 2004, la satisfacción de los usuarios con los motores de búsquedas alcanza 80 puntos de un máximo de 100 [1].

Junto con el éxito en el crecimiento de Internet se ha verificado un aumento del número de sitios de intranets, es decir, de sitios web intra-organizacionales separados de Internet por firewalls o proxies, lo que ha generado una necesidad de motores de búsqueda con características especiales para intranets. Sin embargo, aún cuando la búsqueda de páginas en Internet ha recibido gran atención académica y comercial, se ha realizado poca investigación sobre formas específicas para realizar búsquedas dentro de un sitio web [5, 16]. Un estudio de *Keynote Systems* publicado en enero del 2005 además de reafirmar la supremacía de Google en las búsquedas de Internet, hace notar la alta frustración de los usuarios (un 22 %) con las búsquedas locales en un sitio [9].

Un informe de Jakob Nielsen del año 2002 [13] señala que las búsquedas en sitios de intranet tienen un pobre desempeño, independiente de si son implementadas con un motor de búsqueda propio o utilizando alguno de los servicios públicos. Los problemas encontrados se deben principalmente a que el conjunto de resultados no es priorizado correctamente y la información en el despliegue de resultados no es suficientemente explicativa para que un usuario encuentre lo buscado. Las causas principales para estas fallas son:

- Grandes grupos de páginas normalmente contienen idénticos títulos y resúmenes aún cuando la información contenida sea diferente, lo que afecta el despliegue de información encontrada en su detalle.
- Los links entre páginas no existen debido a la importancia de la información de una página, sino que principalmente por motivos de navegación y estructura del sitio, lo que afecta los algoritmos de rankeo de páginas.
- El contexto en el cual una página existe y las relaciones con otras páginas no puede ser vista a través de la visualización estándar de resultados [5].

Estos problemas dificultan la utilización de las intranets y hacen perder tiempo a los usuarios en búsquedas ineficaces. Una mejora en las intranets, en su diseño, forma de navegación y búsquedas podría disminuir estas pérdidas de tiempo y dinero en hasta un 43 % [13].

El presente trabajo utiliza conceptos de la Web Semántica con el objetivo de obtener mejores resultados que los que se obtienen actualmente al utilizar búsquedas tradicionales en un sitio de intranet. Para esto, se propone un modelo para la catalogación semi-automática del sitio, creando un conjunto de metadatos sobre los contenidos de los recursos disponibles en un sitio, los que son organizados formando un *catálogo*. Los visitantes del sitio web realizan diferentes tipos de consultas en el *servidor de catalogación* para encontrar el o los recursos del sitio que responden a sus necesidades. Los metadatos son creados según un esquema particular del sitio y son ingresados, revisados y mantenidos por un grupo de usuarios catalogadores que agregan a su navegador una herramienta especializada llamada el *cliente de catalogación*.

Se desarrolló una implementación de este modelo llamada *Sistema Catálogo*[1]. Éste se compone de un conjunto de aplicaciones web para búsqueda de recursos, mantención de metadatos y definición de esquemas de metadatos, y de un plugin para navegadores basados en Mozilla. Se realizó además un caso de estudio con la instalación de este sistema para un sitio web corporativo. Para esto se creó un esquema de metadatos particular al sitio, se efectuó una catalogación automática y manual, y luego se realizaron búsquedas de pruebas para evaluar las características y los resultados de este modelo.

Al utilizar el sistema de catalogación para realizar búsquedas se verificó que la cantidad de resultados encontrados es menor que la cantidad que se puede encontrar con un buscador sintáctico, siendo el primero

[1] http://putu.dcc.uchile.cl/catalogo/

normalmente el conjunto de los resultados más relevantes para la búsqueda realizada. El sistema permite realizar búsquedas similares a las de un buscador tradicional, siendo posible además restringir el contexto de las palabras. Se pueden realizar también búsquedas de recursos asociados con alguna instancia de clase (por ejemplo, con alguna persona en particular) o de páginas que referencien a cierto recurso.

Este modelo de catalogación permite, además, utilizar metadatos en la web actual sin necesidad de modificar las páginas web publicadas, lo que permite comenzar a utilizar la Web Semántica sin necesidad de intervenir cada sitio ya existente. Es importante notar también que el catálogo es extensible y puede tener diferentes usos aparte de realizar búsquedas de recursos disponibles en el sitio.

A continuación se presenta el contexto en el cual se desarrolló este trabajo y las investigaciones que lo han influenciado. En la sección 2 se explican detalles generales sobre la catalogación y se presenta el modelo de catalogación propuesto. En la sección 3 se presenta el software desarrollado con la implementación del modelo de catalogación. La sección 4 muestra un caso de estudio con la instalación del sistema para un sitio corporativo. Finalmente la sección 5 muestra los resultados de la catalogación aplicada a un sitio web y se presenta una comparación entre ésta y un buscador tradicional en un sitio web.

1.1. Trabajo relacionado

Este trabajo se enmarca dentro de los proyectos realizados por el Grupo Metadatos de la Universidad de Chile para avanzar hacia la Web Semántica y está basado en la presentación de Tesis de Magíster del año 2003 *Catalogación semántica de sitios web* [12]. Entre los proyectos realizados por el Grupo Metadatos, el presente trabajo se relaciona con *DepMark* [10] al utilizar parte de su ontología para la instalación del sistema de catalogación en el sitio web del Departamento de Ciencias de la Computación de la Universidad de Chile (DCC).

El problema de los buscadores sintácticos relacionado con el despliegue de resultados es tratado por el sistema *Cha-Cha* [5] de la Universidad de California. Este sistema propone un cambio en el diseño de la visualización de los resultados de búsqueda para permitir ver el contexto temático de cada página. Sin embargo, este contexto es deducido según los directorios contenidos dentro de la URL que tiene asignada cada página y no como información que ha sido agregada por humanos, como lo propone la Web Semántica.

El proyecto *Annotea* [19, 8] de la W3C tiene por objetivo permitir la creación y publicación de comentarios sobre documentos web utilizando un esquema basado en RDF y XML. Estas anotaciones son acumuladas en servidores centrales y son ingresadas y visualizadas por programas clientes creados o adaptados para ello. Un programa cliente de este proyecto es *Annozilla*[2], el cual es un plugin para el navegador Mozilla que permite ingresar anotaciones para las páginas web que se estén visitando. El presente trabajo utilizó el código fuente de Annozilla como base para la implementación del cliente de catalogación.

Los Directorios Web intentan lograr una catalogación global de Internet clasificando todos los sitios disponibles en la Web a través de un árbol temático universal. La clasificación es realizada y mantenida por personas -ya sea voluntarios o contratados- que asignan manualmente cada sitio en uno o más grupos dentro del árbol. Los directorios más utilizados en la actualidad son *Open Directory Project*, *Yahoo! Directory* y *LookSmart*[3]. Sin embargo, los Directorios Web no han tenido impacto como una forma para efectuar búsquedas en Internet a través de un catálogo, sino que han tenido mayor efectividad como apoyo a los buscadores sintácticos, presentando temas y sitios relacionados a las palabras buscadas [4].

El enfoque más utilizado actualmente en Internet para usar metadatos es agregar etiquetas `<META>`, también llamadas *meta-etiquetas*, a un documento HTML, con información relevante del texto como palabras claves, título, descripción, tiempo de actualización, etc. Las meta-etiquetas tienen el objetivo de guiar a los buscadores sintácticos en la indexación, búsqueda de resultados y medición de relevancia. Sin embargo, principalmente debido al mal uso dado a este tipo de información extra, los mayores motores de búsqueda han disminuido, y algunos eliminado, el soporte para las meta-etiquetas, con lo cual cada vez pierden mayor importancia, al menos en el ámbito de la búsqueda tradicional [17]. Por esta razón y a la infactibilidad de modificar la web existente para agregar meta-etiquetas, es que el presente trabajo apoya el uso de catálogos sobre sitios web como opción para el uso de metadatos en Internet.

Existe una gran variedad de software de catalogación para la implementación de bibliotecas digitales, sin embargo el presente trabajo no tiene por objetivo final la catalogación formal de recursos en Internet, sino que uno más pragmático como es la mejora de búsquedas en sitios de intranet. En este aspecto el presente

[2] http://annozilla.mozdev.org

[3] http://www.dmoz.org, http://dir.yahoo.com y http://www.looksmart.com, respectivamente.

trabajo se aleja -en principio- de las bibliotecas digitales y sus proyectos relacionados y de los softwares de representación del conocimiento.

La compañía australiana *Metabrowser Systems*[4] ha desarrollado dos productos comerciales para la creación de depósitos de metadatos. El primero es un navegador de Internet basado en MS IE que permite ver las meta-etiquetas de las páginas visitadas y crear nuevos metadatos según diferentes esquemas de metadatos, entre ellos Dublin Core. El segundo es un software repositorio de metadatos que ha sido liberado recientemente como software de prueba. Está basado en tecnología *.NET* y permite contener y administrar los metadatos ingresados. Metabrowser utiliza una arquitectura similar a la propuesta por el presente trabajo, aunque este último la utiliza con el objetivo de mejorar las búsquedas en una intranet proporcionando el software libre necesario para lograrlo.

1.2. Contribuciones

Las principales contribuciones de este trabajo son las siguientes:

- Presentar y detallar una opción para mejorar los resultados que se obtienen actualmente al utilizar búsquedas sintácticas en un sitio de intranet que utiliza técnicas de Web Semántica.
- Proporcionar la implementación del software libre que permite mantener catálogos de recursos en la Web y de una herramienta de catalogación en la forma de plugin de navegador.
- Realizar una instalación piloto en un sitio particular, a partir de la cual se realizaron pruebas y obtuvieron conclusiones sobre el modelo propuesto, comparándolo con un buscador tradicional.

2. Catalogación de un sitio web

Catalogar corresponde al proceso de crear un registro substituto o *metadato* para grupos de información como libros, vídeos, discos, sitios web, etc. [2]. El conjunto de registros conforma un *catálogo* y cumple con tres funciones básicas:

- Conocer qué recursos hay disponibles.
- Conocer dónde se encuentra cada uno de estos recursos.
- Reunir recursos relacionados.

Para el caso particular de documentos electrónicos, el proceso de creación de metadatos se puede definir como la actividad que consiste en extraer y añadir información sobre documentos publicados en aras de su posterior recuperación o para incrementar su utilidad. Se trata, por tanto, de un *arte* de índole técnico, ya que requiere cierta destreza y conocimiento de lenguajes de marcado, formato en que está realizada la publicación electrónica, así como del estándar de metadatos aplicable a tal efecto y del sistema de búsqueda utilizado [11]. El responsable de la creación de los metadatos puede ser el autor del documento mismo, en el caso de bibliotecas digitales tenderán a ser personas especialistas en técnicas de catalogación, y en el caso de sistemas de información especializados los catalogadores deben manejar además los detalles de los conceptos específicos involucrados.

Para el caso de este trabajo, la catalogación es realizada por un conjunto de usuarios que deben tener conocimientos básicos en la herramienta de catalogación y del esquema de metadatos usado. Los autores de los documentos pueden ser -y normalmente los son- usuarios catalogadores del sistema, donde pueden ingresar o actualizar los metadatos de los documentos existentes en el sitio.

Uno de los modelos más utilizados en la actualidad para la creación de la información agregada de un documento es el *Resource Description Framework* (RDF), que permite expresar afirmaciones del tipo: un *recurso* tiene una *propiedad* con un cierto *valor*. Por tanto, sus sentencias son tripletas de la forma sujeto-predicado-objeto, donde *sujeto* puede ser, por ejemplo, una persona, una página web, etc.; *predicado* puede ser la relación "es autor de", "es hermano de", etc.; y *objeto* puede ser un libro, otra persona, etc. Las sentencias de RDF son representadas mediantes grafos dirigidos, donde el sujeto tiene un arco hacia

[4] http://metabrowser.spirit.net.au/

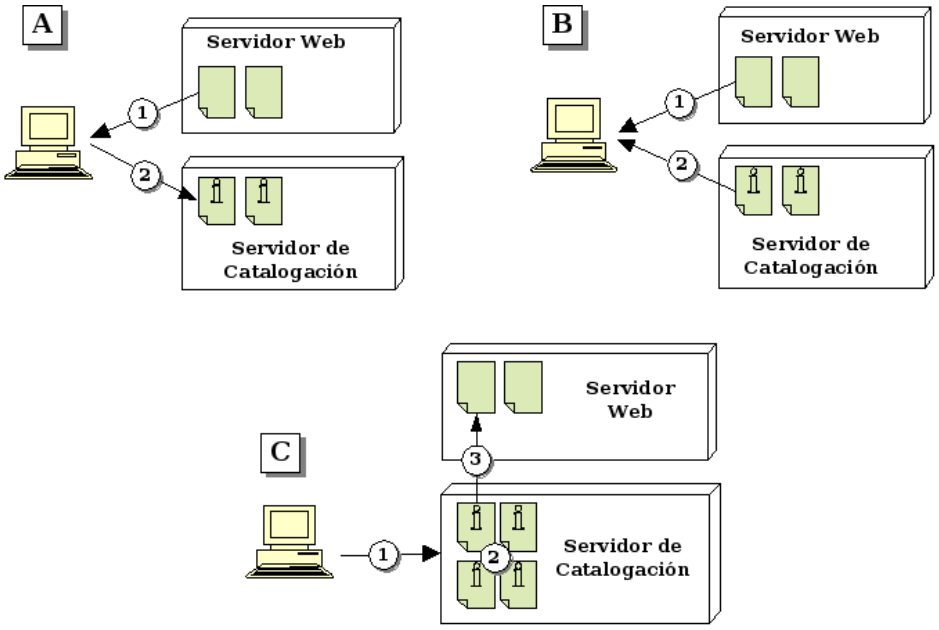


Figura 1: Acciones del sistema de catalogación. **A.** *Agregar metadatos.* 1. Ver Página. 2. Asignar metadatos a URL. **B.** *Apoyar la navegación.* 1. Ver Página. 2. Ver metadatos de URL. **C.** *Buscar recursos.* 1. Ingresar condiciones de búsqueda. 2. Búsqueda entre metadatos. 3. Redirigir hacia URL de los metadatos encontrados.

el objeto mediante el predicado. Un grupo de sentencias forman un grafo RDF, el cual contiene todas las relaciones existentes entre los elementos involucrados en las sentencias [18].

En tareas de catalogación de páginas web, RDF permite describir los contenidos mediante sentencias que contienen al recurso web como sujeto y los predicados corresponden a cada uno de los aspectos a registrar, los que son definidos mediante *ontologías*. Una ontología es la especificación de un vocabulario para un dominio común, es decir, es un modelo para el registro de información, que puede ser definido utilizando RDF a través del lenguaje *RDF Schema*, o utilizando un lenguaje especializado para crear ontologías llamado *OWL*.

Entre los modelos comúnmente usados para registrar datos en Internet se ha destacado el *Dublin Metadata Core Element Set*[5]. Dublin Core es una lista básica de quince elementos diseñada para que los autores y publicadores de documentos de Internet puedan crear sus propios registros sin gran entrenamiento previo.

Para este trabajo, se plantea la creación de un esquema de metadatos específico al sitio en catalogación diseñado pensando en las búsquedas que se desee mejorar, o en su defecto, la utilización de un esquema de metadatos genérico basado en Dublin Core.

2.1. Modelo de catalogación

El modelo propuesto para el sistema de catalogación está compuesto de dos elementos: **el servidor de catalogación** y **el cliente de catalogación**. El servidor de catalogación se encarga de la persistencia de la información del catálogo y de publicar aplicaciones para que los diferentes usuarios puedan hacer uso del catálogo. Permite definir el esquema de metadatos a utilizar, crear y mantener metadatos de los recursos catalogados y presenta diferentes tipos de buscadores y navegadores de metadatos. El cliente de catalogación corresponde a una herramienta que reside en el computador del usuario como un plugin para un navegador. Permite visualizar los metadatos asignados a cada página que se esté visitando, y en el caso que el usuario tenga los permisos necesarios, permite agregar y modificar metadatos en el catálogo.

La figura 1 resume los tres posibles usos que permite el modelo de catalogación:

- Agregar metadatos (figura 1-A), que corresponde a crear un registro para cierta página y agregarlo al catálogo. El ingreso del metadato se puede realizar ya sea utilizando el cliente de catalogación o una aplicación adecuada en el servidor de catalogación.

[5]http://dublincore.org/

- Apoyar la navegación (figura 1-B), que corresponde a obtener los metadatos en el catálogo de cierta página que se esté visualizando, utilizando el cliente de catalogación.
- Buscar recursos (figura 1-C), que corresponde a realizar búsquedas o navegaciones en el catálogo para encontrar páginas con la información requerida.

2.2. Roles de usuario

En el sistema de catalogación se diferencian cuatro diferentes roles involucrados, cada uno con diferentes tareas y responsabilidades:

Interesado o Dueño del sitio Corresponde a la persona u organización que desea tener un catálogo para mejorar las búsquedas en su sitio. Las labores principales que realiza en el sistema son: definir los límites del sitio a catalogar; asignar los usuarios Catalogadores del sistema; y proponer posibles necesidades de interfaz gráfica en los buscadores de recursos.

Administrador Corresponde al usuario experto en el sistema de catalogación. Sus principales acciones en el sistema son: estudiar el sitio a catalogar, su nivel de estructuración, su ámbito temático y sus características particulares; definir el o los esquemas de metadatos a utilizar en el sistema, es decir, debe decidir los metadatos a capturar de una página, las definiciones de clase que contendrá el esquema y definir los valores de referencia a utilizar; realizar una carga inicial de metadatos de las páginas creando un *script* que ingrese masivamente metadatos para la mayor cantidad de páginas posibles; y monitorear el sistema a través de indicadores generales con el objeto de revisar la calidad de los metadatos una vez que el sistema esté en funcionamiento.

Catalogador Corresponde al usuario encargado de mantener los metadatos en el catálogo. Normalmente corresponde a una gran cantidad de personas que deben tener conocimiento básico sobre catalogación de páginas, la herramienta de mantención de metadatos e instancias, y el esquema de metadatos usado en el sitio. Sus labores principales en el sistema son: crear, revisar y mantener los metadatos existentes en el sistema; crear, revisar y mantener las instancias de las clases existentes en el sistema; y recibir y procesar las notificaciones de metadatos erróneos en una página.

Público General o Visitantes Corresponde al usuario que visita el sitio y utiliza el sistema para buscar recursos en él. No tiene conocimiento previo del sistema. Sus acciones son: realizar búsquedas de recursos utilizando alguno de los diferentes buscadores que provee el sistema; navegar el sitio y, en el caso de contar con el cliente de catalogación, puede obtener los metadatos de una página como guía para su navegación; y notificar posibles datos erróneos o imprecisos existentes en el catálogo.

2.3. Esquema de metadatos

Para poder ingresar metadatos dentro del catálogo, es necesario definir previamente el o los esquemas de metadatos que se utilizarán. La definición del esquema de metadatos está representado como un conjunto de tipos de campo organizados en forma de árbol donde cada uno debe tener un elemento padre. Existen cinco posibles tipos de campo (ver figura 2):

- **Ontología**. Es el campo raíz de un esquema de metadatos, permite agrupar los campos en una sola unidad temática. Existen tantos esquemas de metadatos como campos tipo Ontología se hayan definido. Cada Campo de Información hijo corresponde a cada uno de los tipos de metadatos que se capturarán de cada recurso del sitio.
- **Definición de Clase**. Permite declarar una clase dentro de la ontología. Debe ser hijo de un campo Ontología o, en el caso que se defina una subclase, otro campo Definición de Clase. Cada Campo de Información hijo corresponde a cada uno de los atributos de la clase que define.
- **Conjunto de Referencia**. Permite declarar un grupo de valores que corresponden a las posibles opciones de un campo de tipo Elección de Referencia. Debe ser hijo de un campo Ontología o, en el caso que se defina un subgrupo, otro Grupo de Referencia. Cada campo tipo Valor de Referencia hijo corresponde a cada una de las opciones posibles de elección.

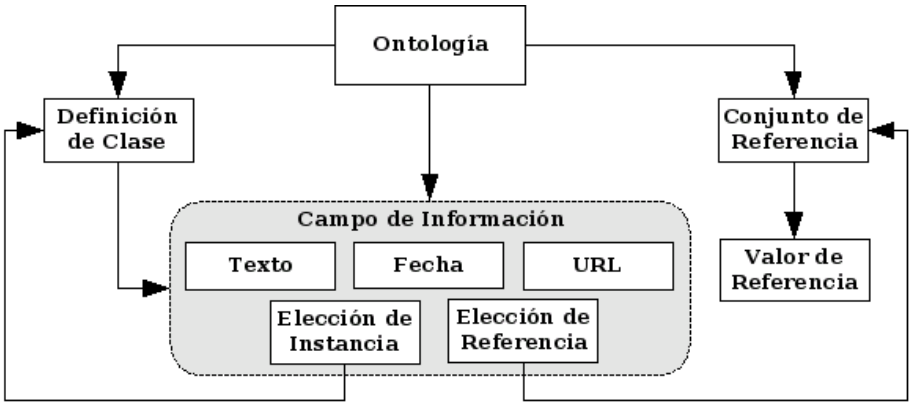


Figura 2: Estructura que cumple cada esquema de metadatos definido en el servidor de catalogación.

- **Valor de Referencia**. Es un campo que representa una opción dentro de un Conjunto de Referencia. Debe ser hijo de un Grupo de Referencia o, en el caso que se defina una especificación, otro Valor de Referencia.
- **Campo de Información**. Es un campo que permite que un usuario catalogador ingrese información sobre cierto elemento. Esta información corresponderá a metadatos de un recurso del sitio en el caso que el campo sea hijo de una ontología, o a atributos de una instancia en el caso que el campo sea hijo de una Definición de Clase. Cada campo de Información puede ser hijo de otro campo de Información para señalar una especificación del campo. Este tipo de campo debe ser uno de los siguientes cinco tipos según el formato aceptable para su valor:
  - *Texto*. Es un campo cuyo valor es un texto sin restricciones.
  - *Fecha*. Es un campo cuyo valor corresponde a una fecha seleccionable de un calendario.
  - *URL*. Es un campo cuyo valor representa una página web. En el caso que la página referenciada exista dentro del servidor esta última aumentará su relevancia base.
  - *Elección de Instancia*. Es un campo cuyo valor debe ser alguna de las instancias creadas para una Definición de Clase definida.
  - *Elección de Referencia*. Es un campo cuyo valor debe ser uno de los Valores de Referencia dentro de un Conjunto de Referencia definido.

2.4. Instalación y puesta en marcha

Para poder contar con el sistema de catalogación en un sitio web se debe realizar una serie de tareas cada una con diferentes responsables:

1. El usuario Interesado debe decidir el tamaño del sitio a catalogar.
2. El usuario Administrador debe estudiar el sitio, la información que contiene y su nivel de estructuración para decidir los recursos que deben ser catalogados. El nivel de estructuración muestra además como se puede realizar la carga inicial de datos, lo que permite estimar los beneficios que se pueden lograr y el tiempo requerido.
3. El usuario Administrador debe definir e ingresar en el servidor de catalogación el o los esquemas de metadatos a utilizar. Para esto se debe decidir los metadatos a ingresar por cada página y las clases y valores de referencia a crear.
4. El usuario Administrador debe ingresar al catálogo las instancias conocidas de antemano para las clases definidas en el esquema de metadatos.

Figura 3: Buscador de recursos catalogados. En el ejemplo se buscan las páginas que posean algún metadato que contenga las palabras “clase” y “auxiliar” y que pertenezcan al curso “CC10A - Computación I”.

5. El usuario Administrador debe hacer una carga inicial de metadatos de las páginas del sitio, que contengan la mayor cantidad de datos del esquema. En lo posible que contengan el título de la página, las referencias entre páginas y las asociaciones con las instancias ya creadas.

6. El usuario Interesado debe decidir quienes cumplirán la labor de usuarios Catalogadores del sistema.

7. Los usuarios Catalogadores deben realizar la catalogación manual de páginas, verificando el marcado automático e ingresando nuevas páginas y metadatos al catálogo.

8. El usuario Administrador debe implementar buscadores particulares al sitio y/o modificar los buscadores genéricos para asemejarse al diseño gráfico del sitio.

9. El usuario Administrador debe hacer ajustes sobre los puntos de rankeo de los campos de metadatos y de las referencias entre páginas, realizando búsquedas de prueba hasta verificar resultados satisfactorios en el rankeo de resultados.

Una vez que el sistema está en funcionamiento, los visitantes del sitio realizan búsquedas de recursos utilizando las aplicaciones correspondientes. En ese momento se inicia el proceso de mantención de metadatos, que comprende las siguientes tareas:

1. Los usuarios Visitantes del sitio, en el caso de encontrar anomalías en los metadatos existentes en el catálogo, notifican los posibles problemas de datos en el sistema.

2. Los usuarios Catalogadores deben realizar mantención periódica de los metadatos en el sistema, agregando metadatos para nuevas páginas, actualizando metadatos para páginas que hayan sido modificadas, o eliminando metadatos de páginas borradas del sitio. Reciben y procesan además las notificaciones de metadatos erróneos recibidas.

3. El usuario Administrador monitorea la calidad de los metadatos del sistema a través de indicadores proporcionados por el servidor de catalogación.

3. Sistema Catálogo

Sistema Catálogo es el sistema de catalogación desarrollado en el marco de este trabajo con el objetivo de demostrar las capacidades del modelo presentado en obtener mejores resultados que los buscadores sintácticos en un sitio web. Se compone del servidor de catalogación y cliente de catalogación descritos a continuación.

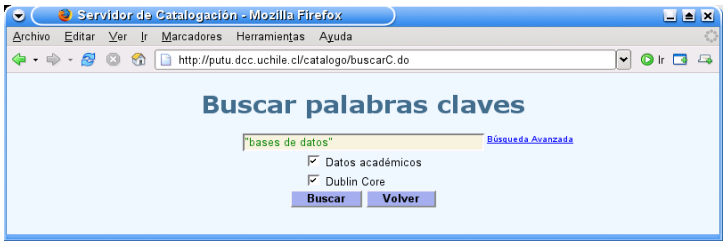


Figura 4: Buscador de recursos por palabras claves. En el ejemplo se buscan las páginas que posean algún metadato que contenga la frase "bases de datos".

3.1. Servidor de catalogación

Es un conjunto de aplicaciones desarrolladas utilizando tecnología Java sobre un servidor web para las implementaciones y una base de datos relacional para la persistencia de los esquemas y metadatos. Contiene tres aplicaciones web: Búsqueda, Catalogación y Administración, las que pueden ser accedidas por un usuario tipo Visitante, Catalogador y Administrador, respectivamente. Los requerimientos técnicos para instalar el servidor de catalogación son los siguientes:

- J2SDK 1.4.
- Jakarta Tomcat 5.0.
- PostgreSQL 7.4.

La figura 3 muestra un buscador de recursos catalogados. El ingreso de los parámetros se realiza definiendo primero la ontología o el campo específico sobre el cual se desea realizar la búsqueda y luego el valor a buscar. Según el tipo del campo seleccionado se despliega la forma de ingresar el valor correspondiente: una lista de instancias para un campo tipo Elección de Instancia, una lista de Valores de Referencia para un campo tipo Elección de Referencia, un calendario para un campo tipo Fecha o un cuadro de texto para un campo tipo Texto Libre o URL. En el ejemplo de la figura se realiza una búsqueda de las páginas cuyo conjunto de metadatos cumpla con dos condiciones simultáneamente: que posea un metadato clasificado bajo la ontología *Datos Académicos* cuyo texto contenga las palabras "clase" y "auxiliar", y que posea un metadato para el campo *Pertenece al Curso* cuyo valor sea una referencia a la instancia "CC10A - Computación I" de la clase Curso.

La figura 4 muestra un buscador de recursos basado sólo en palabras claves. Su interfaz es similar a un buscador sintáctico tradicional, permite ingresar un texto a buscar entre los metadatos definidos para las páginas catalogadas, restringiendo el universo de búsqueda a los metadatos asignados bajo una o más ontologías. En el ejemplo de la figura se realiza una búsqueda de las páginas que contengan en alguno de sus metadatos la frase "bases de datos". En el caso de la búsqueda avanzada, se pueden realizar múltiples búsquedas de palabras restringiendo cada una a los metadatos asignados bajo una ontología o algún campo específico de ésta.

3.2. Cliente de catalogación

Es una herramienta desarrollada utilizando tecnología de Mozilla (XUL y JavaScript), que puede ser agregada como barra lateral del navegador. El cliente está empaquetado en un archivo XPI lo que permite que sea instalado automáticamente dentro del navegador. Una vez instalado el cliente se debe agregar una referencia en la barra lateral del navegador a la dirección *chrome://cliente/content/panel.xul*. Los requerimientos técnicos para instalar el cliente es utilizar un navegador Mozilla o Mozilla-Firefox 1.0 o mayor, que tenga habilitada la capacidad de instalar software.

El plugin permite dos tareas: Ver metadatos y Modificar metadatos, las que pueden ser accedidas por un usuario tipo Visitante y Catalogador, respectivamente.

La figura 5 muestra el navegador con la barra lateral del cliente de catalogación. Al presionar el botón *Refresh*, se realiza una consulta al servidor de catalogación por los metadatos correspondientes a la URL de la página que se encuentra en la ventana central del navegador, los que son desplegados en forma de árbol.

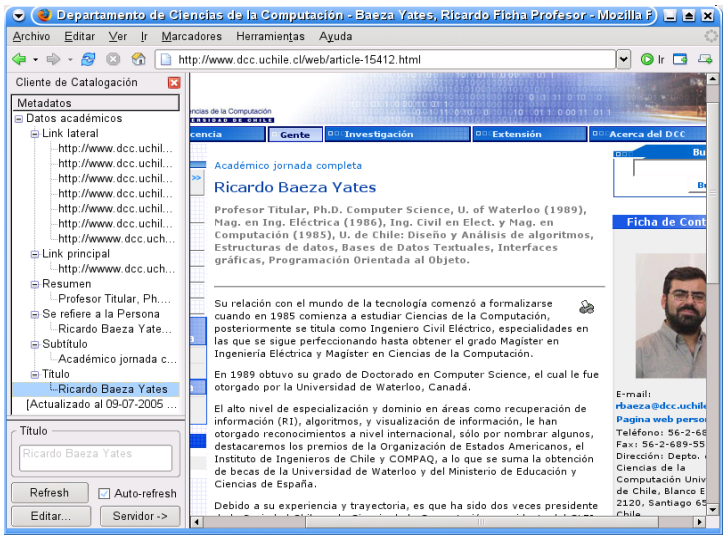


Figura 5: Visualización de los metadatos de un recurso catalogado utilizando el cliente de catalogación.

Al marcar el recuadro *auto-refresh* se habilita el modo automático, donde cada 10 segundos se realiza la acción del botón *Refresh*.

4. Caso de estudio

Como prueba del sistema, se realizó la catalogación del sitio web del Departamento de Ciencias de la Computación de la Universidad de Chile (DCC), cuya URL es *http://www.dcc.uchile.cl*.

Se utilizó un esquema de metadatos particular al sitio, el cual es el resultado del estudio del conjunto de páginas a catalogar y de la ontología utilizada por el proyecto *DepMark* [10]. Contiene cuatro campos de tipo Texto: *Título*, *Subtítulo*, *Resumen* y *Sección*; dos campos de tipo URL: *Link principal* y *Link lateral*; y tres campos de tipo Elección de Instancia: *Pertenece a la Carrera*, *Pertenece al Curso* y *Se refiere a la Persona* que referencian a instancias de las clases *Carrera*, *Curso* y *Persona*, respectivamente. Además del esquema particular al sitio, se ingresó en el sistema el esquema de datos definidos por Dublin Core para sus elementos básicos y sus calificadores.

Para realizar la carga automática primero se hizo una copia local del sitio web publicado. Se recolectaron 15 MB en 835 archivos, de las cuales sólo se seleccionaron 353 páginas para ser catalogadas con un espacio total de 1 MB. Las páginas restantes no fueron catalogadas por ser páginas de enlace, que sólo presentan resúmenes de otras o son versiones imprimibles.

Luego se procedió a implementar un conjunto de *scripts* para procesar el texto del HTML utilizando expresiones regulares. Se capturaron los campos de tipo Texto y URL, y se insertaron en la base de datos del sistema de catalogación. La creación de instancias fue manual y correspondió a un total de 99 instancias: 4 carreras, 51 cursos y 44 personas. Después se realizaron, también manualmente, las asociaciones entre instancias y páginas a través de los campos de tipo Elección de Instancia.

El conjunto de metadatos ingresados finalmente, sumando metadatos de páginas y atributos de instancia,

	Tarea	***t***
1.	Definición del conjunto a catalogar y estudio de la estructura del sitio.	2 días
2.	Definición el esquema de metadatos.	2 días
3.	Ingreso de instancias en forma manual.	1 día
4.	Carga inicial de metadatos.	2 días
5.	Marcado manual del sitio.	7 días
6.	Pruebas de búsquedas y ajuste de los puntos de rankeo del esquema.	2 días
	Total	*16 días*

Cuadro 1: Resumen del tiempo para la puesta en marcha del sistema.

Aspecto	Catalogación de un sitio	Buscador sintáctico de un sitio
Costos	Mayor cantidad de trabajo y tiempo para lograr catalogar un sitio. Necesidad de un experto en el sistema de catalogación.	Baja cantidad de tiempo y conocimientos necesarios para tener el sistema en funcionamiento.
Mantención	Requiere de un usuario administrador para monitorear el estado del sistema y de usuarios catalogadores para verificar y actualizar los metadatos.	Requiere de un usuario administrador para monitorear el estado del sistema.
Resultados	Menor cantidad de resultados encontrados, pero los encontrados son de mayor relevancia para la búsqueda.	Al encontrar todas las páginas donde se encuentra cierta palabra, normalmente los resultados son una gran cantidad de páginas muy similares.
Formas de búsqueda	Diferentes tipos de buscadores que pueden ser usados según la cantidad de información que se tenga sobre lo buscado. En el caso de tener pocos conocimientos se puede intentar una navegación del catálogo.	Interfaz simple de búsqueda. Poca utilización de las búsquedas avanzadas. Difícil de utilizar en el caso de tener poco conocimiento en el área buscada.
Contexto de resultados	Permite conocer el contexto de cada página, independiente de la forma de navegación.	No existe forma de conocer el contexto de una página.
Recursos indexados	Se puede agregar al catálogo todo tipo de documento, incluido cualquier archivo binario.	Se pueden indexar archivos de texto y archivos binarios que puedan ser transformados automáticamente en texto.
Uso de metadatos	Permite hacer uso de metadatos en la web sin necesidad de modificar la web existente.	Para hacer uso de metadatos requiere de la modificación de las páginas web ya publicadas para agregar las meta-etiquetas correspondientes.

Cuadro 2: Comparación entre el sistema de catalogación y los buscadores sintácticos de un sitio.

fue un total de 2440 los que ocuparon un espacio aproximado de 104 KB (tamaño en bytes de los textos de todos los metadatos). Por lo cual, los metadatos correspondieron a aproximadamente un 10 % del tamaño de las paginas catalogadas y a un 0,7 % del tamaño total del sitio.

El cuadro 1 resume el tiempo tomado para que el sistema de catalogación haya quedado disponible para realizar búsquedas al público. El esfuerzo realizado fue de aproximadamente tres semanas por una sola persona. Se verificó que la tarea que toma mayor tiempo es la marcación manual del sitio. Una de las razones que incidió en esto fue que la carga inicial de metadatos no incluyó asociaciones con instancias las cuales debieron ser enlazadas en forma manual.

El sistema se encuentra actualmente disponible para uso público en la dirección web *http://putu.dcc.uchile.cl/catalogo/*.

5. Conclusiones

Al utilizar el sistema de catalogación para realizar búsquedas se verifica que la cantidad de resultados encontrados es menor que la cantidad que se puede encontrar con un buscador sintáctico, correspondiendo normalmente al conjunto de los resultados más relevantes para la búsqueda realizada.

Al desarrollar el buscador de metadatos quedó de manifiesto que en un catálogo no puede existir sólo una interfaz de búsqueda, sino que debe permitir múltiples y variadas formas para realizar consultas. Es de esperarse que en un principio el tipo de buscador más utilizado sea el presentado en la figura 4 por su mayor similitud con un buscador sintáctico. Sin embargo, una vez que los usuarios adquieren conocimiento sobre el catálogo y sus capacidades, se hace más factible utilizar alguno de los navegadores o buscadores proporcionados que hacen mayor uso del potencial de los metadatos.

Desarrollar una interfaz genérica para los diferentes buscadores es un problema que no pudo ser solucionado satisfactoriamente. Se intentó realizar una interfaz más amigable incluso disminuyendo la potencia del buscador (permitiendo ingresar búsquedas anidadas en un solo nivel cuando el motor permite múltiples niveles), sin embargo el problema aún está abierto para mejores soluciones genéricas. Se puede afrontar este problema implementando buscadores especializados para cada sitio en particular que se adecuen al diseño de éste y a sus esquemas de metadatos.

Una característica importante del sistema de catalogación es que no es necesario modificar las páginas

web existentes para hacer uso de él. Esto permite la utilización de metadatos para mejorar las búsquedas sin necesidad de modificar un sitio ya existente.

La información contenida en el catálogo además es extensible para otros posibles usos independientes del buscador semántico. Por tanto, el catálogo es un recurso que tiene gran potencial y sirve de base para desarrollar futuras ideas y proyectos relacionados con la Web Semántica, como por ejemplo realizar mapas de sitio o implementar búsquedas inter-sitios.

A modo de resumen, el cuadro 2 presenta una comparación entre el modelo de catalogación propuesto y los buscadores sintácticos de un sitio, según diferentes aspectos.

6. Trabajo futuro

- Estudiar los problemas asociados a la unicidad de una página según su URL. En particular estudiar los casos de páginas dinámicas, páginas por defecto y mirrors de sitios.
- Estudiar el problema de la interfaz de usuarios para un buscador semántico. Debe ser lo suficientemente poderosa para permitir el ingreso de consultas complejas, pero debe permitir que un usuario promedio sin conocimiento previos pueda hacer uso de ella.
- Estudiar formas de integración entre catálogos, y por consiguiente, estudiar la integración entre diferentes esquemas de metadatos.
- Estudiar el uso del sistema de catalogación como depósito de referencias de Internet. Se puede utilizar el mismo software de catalogación con el objeto de organizar en forma colaborativa un conjunto muy grande de referencias a páginas de Internet. Se ingresan datos para cada página de interés según cierto esquema de metadatos y luego se realizan consultas sobre estos metadatos para encontrar los links relevantes entre el conjunto de referencias.
- Estudiar el *perfilamiento* en la web. Al poder realizar búsquedas restringiendo los ámbitos de interés a ciertos temas en particular (por ejemplo, artes) y cada página ser catalogada según una persona con el perfil del área (por ejemplo, un artista), se pueden realizar búsquedas y navegar la web bajo cierta forma de ver la información, es decir bajo un perfil particular.

Agradecimientos

Los autores agradecen financiamiento al Proyecto FONDECYT 1030810, "Metadatos para describir y consultar la Web Oculta". Claudio Gutiérrez agradece también al Nucleo Milenio, Centro de Investigación de la Web, P04-067-F, Mideplan.

Referencias

[1] American Customer Satisfaction Index. *Second Quarter Scores: Manufacturing/Durable Goods & E-Business: Search Engines*, Agosto 2004. http://www.theacsi.org.

[2] Susan Atkey. *Issues in Cataloguing the Web*. School of Library, Archival and Information Studies/UBC, Diciembre 2002.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. *The Semantic Web*. Scientific American, Inc, Mayo 2001.

[4] Fidel Cacheda and Angel Viña. *Understanding how people use search engines: a statistical analysis for e-Business*. Proceedings of the e-Business and e-Work Conference and Exhibition (e-2001), Venice, Italy, Octubre 2001.

[5] Michael Chen, Marti Hearst, Jason Hong, and James Lin. *Cha-Cha: A System for Organizing Intranet Search Results*. Proceedings of the 2nd USENIX Symposium on Internet Technologies and SYSTEMS (USITS), Octubre 1999.

[6] Lorcan Dempsey and Rachel Heery. *A review of metadata: a survey of current resource description formats*. UKOLN Metadata Group, Marzo 1997.

[7] Jeff Heflin, James Hendler, and Sean Luke. *SHOE: A Blueprint for the Semantic Web*. Data and Knowledge Engineering. Spinning the Semantic Web. MIT Press, Cambridge, Marzo 2003.

[8] José Kahan and Marja-Riita Koivunen. *Annotea: An Open RDF Infrastructure for Shared Web Annotations*. World Wide Web Consortium, Mayo 2001.

[9] Keynote Systems. *Yahoo! Search and MSN Search Close the Gap with Google. Press Release 05-01-13*, Enero 2005. http://www.keynote.com.

[10] Ernesto Krsulovic Morales and Claudio Gutiérrez. *Building Yearbooks with RDF*. Centro de Investigación de la Web. Departamento de Ciencias de la Computación. Universidad de Chile, Diciembre 2002.

[11] Eva Mª Méndez. *Metadatos y recuperación de información: Estándares, problemas y aplicabilidad en bibliotecas digitales*. Departamento de Biblioteconomía y Documentación de la Universidad Carlos III de Madrid. Ediciones Trea. ISBN: 84-9704-055-4, Junio 2002.

[12] Juan Manuel Barrios N. *Presentación Tema de Tesis para Magister en Ciencias, mención Computación: Catalogación semántica de sitios web*. Departamento de Ciencias de la Computación. Universidad de Chile, Diciembre 2003.

[13] Jakob Nielsen. *Intranet Usability: The Trillion-Dollar Question*. Useit.com Alertbox, Noviembre 2002.

[14] Jakob Nielsen. *When Search Engines Become Answer Engines*. Useit.com Alertbox, Agosto 2004.

[15] Natalya Fridman Noy and Deborah L. McGuinnes. *Ontology Development 101: A Guide to Creating Your First Ontology. Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*. Stanford Knowledge Systems Laboratory, Marzo 2001.

[16] Dick Stenmark. *A Methodology for Intranet Search Engine Evaluation*. In Käkölä, T. (ed.), Proceedings of IRIS22, August 7-10, Department of CS/IS, University of Jyväskylä, Finland, Agosto 1999.

[17] Danny Sullivan. *Death Of A Meta Tag*. Search Engine Watch, Octubre 2002.

[18] World Wide Web Consortium. *Resource Description Framework Model and Syntax Specification. W3C Recommendation*, Febrero 1999.

[19] World Wide Web Consortium. *Annotea Protocols. W3C Draft*, Diciembre 2002.

BDNG – Una propuesta para el diseño e implementación de Bibliotecas Digitales: BDEAFIT un caso práctico.

Edwin N. Montoya, Jorge I. Giraldo, Gloria P. Ospina, Cástulo Ramirez,
Universidad EAFIT, Departamento de Informática y Sistemas
Medellin, Colombia, 3300
{emontoya,jgiral43,gospina,cramir23}@eafit.edu.co

Maryem A. Ruiz
Politécnico Jaime Isaza Cadavid, Departamento de Informática
Medellin, Colombia
mruiznu1@elpoli.edu.co

Abstract

Digital libraries will be one of the main ways to access structured digital information. The use of Digital Libraries has been triggered by the wide deployment of Internet which allows the connectivity and "anywhere" access to content as well as digital resources inside modern libraries (e-books, full-text databases, digitalization process, etc). Even though several digital libraries architectures have been proposed, they have not been widely deployed. This paper presents a proposal of a digital library architecture that has been successfully implemented at EAFIT University. It imports any referential catalogue from any legacy *Library Information System*, adds digital content to referential catalogue or creates new objects in the digital library, provides an interface to search and retrieval and integration with other digital libraries . It has been implemented by using open and standard software, using metadata models based on *Resource Description Framework* (RDF) – key piece for *Semantic Web* – and the *Dublin Core Metadata Initiative* (DCMI) widely accepted in the digital libraries communities.

Keywords: Digital Libraries, Dublin Core, RDF, Services, Metadata.

Resumen

Las bibliotecas digitales serán una de las principales formas de acceder información estructurada. El uso de las bibliotecas digitales ha sido motivado por el amplio despliegue de Internet facilitando la conectividad y acceso al contenido desde cualquier lugar así como a los recursos digitales dentro de las bibliotecas modernas (libros electrónicos, bases de datos de texto completo, procesos de digitalización, etc.) Aunque se han propuesto muchas arquitecturas para el desarrollo de bibliotecas digitales, no se han desplegado masivamente como se esperaría. Este artículo presenta una propuesta de arquitectura para una biblioteca digital que ha sido implementada exitosamente en la Universidad EAFIT. Permite importar cualquier catálogo referencial de cualquier *Sistema de Información Bibliotecarios*, adicionar contenido digital a dicho catálogo o crear nuevos objetos en la biblioteca digital, integrarse con otras bibliotecas digitales y finalmente facilitar al usuario una interfaz de búsqueda y recuperación. Se ha implementado con software estándar y abierto, utilizando propuestas de metadatados basados en *Resource Descripción Framework* (RDF) base fundamental para la *Web Semántica* y la *Iniciativa de Metadatos Dublín Core* modelo ampliamente aceptado en el comunidad de bibliotecas digitales.

Palabras claves: Bibliotecas Digitales, Dublín Core, RDF, Servicios, Metadatos

1. INTRODUCCION

Una *biblioteca digital* (BD) es una colección de recursos digitales los cuales pueden estar distribuidos a través de una red como Internet y organizada a través de un modelo de metadatos. La BD ofrece un conjunto de servicios como gestión, búsqueda y recuperación de información a sus usuarios [1]. En la actualidad las bibliotecas digitales juegan un papel crucial en el establecimiento de enclaves de materiales confiables, en gran medida debido al universo creciente y casi infinito de materiales disponibles a través de la interconexión de redes mundiales. Es por eso que las bibliotecas digitales representan una nueva infraestructura y un nuevo ambiente creado por la integración y el uso de la computación, las comunicaciones, y el contenido digital a escala global, convirtiéndose en parte importante de la infraestructura de la información en este nuevo siglo. Las bibliotecas digitales modernas han llegado a ser el medio para preservar colecciones universales de conocimiento, y un mecanismo de fomento de creatividad y el ingenio para las generaciones futuras, esto basado en el desarrollo de las más recientes investigaciones y tecnologías [2]. Las

bibliotecas digitales contribuyen al uso creciente de la información distribuida en la red, de todas las clases, formas y representaciones.
Una arquitectura para una biblioteca digital considera una serie de aspectos base como: (1) captura o creación de contenido, (2) indexación y catalogación (metadatos), (3) almacenamiento (repositorio), (4) búsquedas/consultas, (5) protección de información y (6) recuperación/distribución [3] [4]. Este artículo describe una propuesta de arquitectura para el diseño e implementación de una biblioteca digital, modular, abierta y estándar que incorpora una base sólida de los componentes propuestos dentro de una arquitectura.
El Grupo *de Investigación en Redes y Sistemas Distribuidos* (GIRSD) de la universidad EAFIT, esta llevando a cabo actividades de investigación y desarrollo en el campo de las bibliotecas digitales desde hace 4 años, con ello, no solo ha contribuido con el desarrollo de este campo, sino que ha desplegado los resultados en la plataforma de producción de la biblioteca digital en dicha universidad y en otros proyectos como el de la integración de bibliotecas digitales y bibliotecas digitales para sistemas de *e-learning*. El proyecto BDEAFIT ha sido desarrollado de una manera incremental a través de proyectos de investigación, tesis de maestría y proyectos fin de carrera. Este artículo está organizado de la siguiente manera: la sección 2, realiza una breve descripción de la arquitectura de BDNG. La sección 3, describe la implementación de la arquitectura BDNG en la Universidad EAFIT (BDEAFIT) como plataforma de biblioteca digital describiendo cada uno de los componentes. La sección 4 describe las líneas de investigación y trabajos futuros y la sección 5 describe las principales conclusiones y contribuciones de este artículo.

2. ARQUITECTURA DE BDNG

La *Biblioteca Digital de Nueva Generación* (BDNG) plantea un marco general de diseño e implementación de bibliotecas digitales, el cual contempla componentes generales como importar catálogos referenciales de otras bibliotecas, manejo de contenido digital, integración con otras bibliotecas digitales y búsqueda y recuperación avanzada de información. Utiliza modelos estándares de metadatos y plantea la implementación con software abierto y estándar lo que facilita su despliegue. BDNG es un marco de trabajo el cual requiere ser particularizado en una aplicación específica. Bajo esta arquitectura se han desarrollado varias bibliotecas digitales como la Biblioteca Digital de la Universidad EAFIT (BDEAFIT), Biblioteca Digital de EAFIT Interactiva (BDEI) para un sistema de *e-learning* y el proyecto de metabiblioteca que consiste en integrar varias bibliotecas referenciales y digitales principalmente de la región metropolitana de Medellín, aunque participan otras bibliotecas nacionales. La figura 1 muestra la visión global del proyecto y la relación con proyectos específicos.

Figura 1. El Proyecto BDNG

BDNG presenta una arquitectura modular, abierta y no intrusiva a los sistemas de información legados de gestión de bibliotecas. Esta compuesta por los siguientes módulos representados en la figura 2:

- Módulo de carga o importación de catálogos (DL:Upload)
- Módulo de búsqueda y recuperación (DL:Search)
- Módulo de adición de contenido digital (DL:AddContent)
- Módulo de creación de contenido digital (DL:CreateContent).
- Módulo de integración (DL:Metalib)
- Módulo de gestión (DL:Management)

Carga de catálogos referenciales (DL:Upload): Este módulo permite importar cualquier catálogo referencial de cualquier Sistema de Información Bibliotecario (SIB) a partir de archivos texto o XML formateados con reglas predefinidas. La carga de los datos en la biblioteca digital se puede realizar en línea o en procesamiento por lotes (*batch*), esto es, el SIB genera los archivos, son transferidos a la BD y son cargados por el administrador del sistema, o puede plantearse que el SIB directamente se comunique con la BD para la carga de datos. Una vez los datos son procesados por este módulo, son almacenados en la base de datos de metadatos en forma de documentos XML.

Búsqueda y Recuperación de información (DL:Search): Este módulo es el encargado de realizar las consultas y de realizar la recuperación de información de interés del usuario. Se plantea dos tipos de recuperación de información, una referida a los metadatos, es decir, el usuario recupera un conjunto de registros de interés y otra de contenido digital en la medida que los registros de metadatos tengan relacionado dicho contenido. La búsqueda se realiza directamente en la base de datos XML, en el repositorio de metadatos. Dichos metadatos podrán contener no solo información descriptiva del contenido físico o digital sino información directamente relacionada con el contenido digital (p.e. búsquedas en texto completo). Se utiliza lenguajes de consulta optimizados para información XML como XPATH y XQUERY así como optimización para búsqueda en texto.

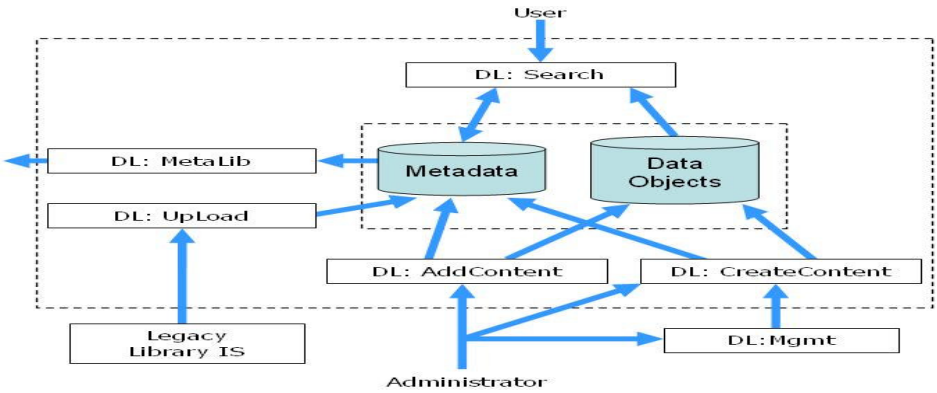


Figura 2. Arquitectura general de BDNG

Adición de contenido digital (DL:AddContent): Este módulo permite asociar y almacenar contenido digital al catálogo referencial proveniente del SIB. Puede manejar tanto contenido digital tradicional como libros, artículos, reportes digitales así como contenido multimedia como video o audio. Permite de manera interactiva, buscar y recuperar un registro sobre el cual se desea trabajar y recuperar el contenido digital (si ya lo tiene). A partir de allí se puede realizar operaciones como adición, modificación o retiro de contenido digital al catálogo. Los objetos digitales son almacenados en dos tipos diferentes de servidores: Servidores Web para almacenar los datos tradicionales y Servidores Multimedia para almacenar el video. Permite asociar metadatos al contenido digital los cuales también son almacenados en la base de datos XML, esto permite enriquecer mucho más los mecanismos de búsqueda por parte de los usuarios.

Creación de contenido digital (DL:CreateContent): Este módulo permite almacenar objetos nuevos en la BD, es decir, permite crear los registros descriptivos de metadatos así como almacenar el contenido digital en los respectivos servidores de acuerdo a su tipo.

Integración con otras bibliotecas (DL:MetaLib): Este módulo permite conectarse como Cliente con otras bibliotecas digitales tanto bajo el mecanismo de comunicación en línea o bajo el mecanismo de recolección como Proveedor de datos bajo el protocolo OAI-PMH [5]. Este módulo también le permite a la BD actuar como un Servidor de Metabiblioteca bajo varios mecanismos: cargando catálogos de otros repositorios, acceso en línea a otros sistemas bibliotecarios o como Recolector o proveedor de servicios en el protocolo OAI-PMH.

Gestión (DL:Management): Este módulo permite realizar todas las operaciones de mantenimiento de la BD como gestión de usuarios, gestión de servicios, gestión de servidores de datos, entre otras.

Contenido Digital

La Biblioteca Digital almacena la información en dos repositorios diferentes: (1) Repositorio de metadatos en el cual se almacenan todos los metadatos de los objetos físicos y digitales, no solo almacena los metadatos descriptivos de dichos objetos, sino que puede almacenar metadatos dependientes del contenido lo que enriquece mucho más el módulo de búsqueda y recuperación. (2) Repositorio de Objetos Digitales, hace referencia al contenido digital en sí mismo el cual se clasifica en dos tipos: objetos digitales tradicionales principalmente basados en texto y objetos multimedia como video y audio.

Metadatos
La definición más simple de metadato es datos acerca de datos o información acerca de información [6]. En el contexto de bibliotecas digitales, los metadatos describen cualquier objeto físico o digital y facilita su organización. BDNG utiliza un modelo de metadatos basado en el *Marco de trabajo de Descripción de Re*cursos (*Resource Description Framework* –RDF) y la *Iniciativa de Metadatos Dublín Core* (*Dublín Core Metadata Initiative* – DCMI) [7]. Estos metadatos son codificados en XML los cuales serán almacenados en el repositorio de metadatos.
RDF es una modelo de descripción de recursos que incorpora otros modelos de metadatos, es decir, más que un modelo de metadatos, es un metamodelo que define una norma para crear modelos o esquemas de metadatos. Unido a XML y relacionado con el contenido de la Web, se ha convertido en pieza fundamental para la Web Semántica.
La Iniciativa de Metadatos del Dublín Core (DCMI) o simplemente DC es un modelo de metadatos originalmente pensado para la descripción de recursos en Internet, pero el cual ha ganado amplia aceptación en otros dominios de aplicación, en particular en las bibliotecas digitales. Unido con propuestas como RDF y XML, configuran una base sólida para la representación de metadatos. En su versión más simple, define un conjunto 15 elementos (DCMES, *Dublín Core Metadata Element Set version 1.1*) [8], agrupados en tres categorías: (1) elementos relacionados con el contenido, (2) elementos relacionados con la propiedad intelectual y (3) elementos relacionados con la temporalidad. La tabla N. 1 presenta un resumen de estos elementos.

Elementos de contenido	Elementos de propiedad intelectual	Elementos de temporalidad
titulo	creador/autor	fecha
materia	editor	tipo
descripción	contribuidor	formato
fuente	derechos de autor	identificador
idioma		
relación		
cobertura		

Tabla 1: Elementos del modelo de metadatos Dublín Core

Para una especificación semántica más precisa, algunos elementos hacen uso de refinamientos o calificativos a fin de facilitar la búsqueda y recuperación de los recursos. Cuando se hace uso de refinamientos y/o del elemento adicional audiencia se conoce como *DCMI Qualified* [7]. Igualmente, se han desarrollado vocabularios particulares en dominios de aplicación vertical, en el caso concreto de las bibliotecas digitales, se ha definido el perfil de aplicación DC-Lib [9], el cual permite relacionar información propia de este dominio. En el caso de BDNG y sus aplicaciones, no se utiliza el DCMI calificado, se utiliza un subconjunto de elementos de DC-Lib y se adicionan nuevos elementos de acuerdo a las aplicaciones concretas.
Un aspecto importante de diseño de la biblioteca digital, es definir que tipo de base de datos y en que formato de metadatos almacenar los metadatos para proveer los servicios de búsqueda y recuperación [10]. Normalmente, puede presentarse los siguientes acercamientos:

1. Utilizar los metadatos DC en XML solo en los procesos de entrada y salida del sistema, pero el almacenamiento y búsquedas realizarlo en bases de datos relacionales.
2. Utilizar los metadatos DC en XML en todas las etapas.

BDNG sigue un acercamiento hibrido, en el cual la entrada, salida, almacenamiento y consulta se realiza en bases de datos XML nativas, pero la gestión del contenido digital en los módulos de adición y creación de contenido digital utiliza bases de datos relacionales temporales previo a la publicación de metadatos.

3. BDEAFIT: UNA IMPLENTACION DE BDNG

La *Biblioteca Digital de la Universidad EAFIT* (BDEAFIT), es una aplicación de BDNG en la cual se han desarrollado todos los módulos propuestos por la arquitectura. Fue diseñada e implementada bajo las siguientes premisas y situaciones: 1) Coexistir con cualquier sistema de información bibliotecario legado. 2) Definir un modelo de metadatos estándar, abierto y de amplio uso. 3) Soportar cualquier tipo de contenido digital, tradicional -textos, documentos, libros, etc. y no tradicionales (video, audio, realidad virtual, etc. 4) Integración de catálogos referenciales y contenido digital en un acercamiento de metabiblioteca o federación. 5) Integrar sistemas basados en igual arquitectura y tecnología, sistemas legados y otras bibliotecas digitales diferentes.

Modelo de metadatos en BDEAFIT
BDEAFIT utiliza dos modelos y a su vez repositorios para los metadatos, un primer esquema que almacena los metadatos descriptivos del material físico o digital y otro repositorio que almacena los metadatos relacionados

directamente con el contenido digital. Esta separación de metadatos permite relacionar varios objetos digitales a un único registro referencial. Por esta razón, algunos de los elementos relacionados con el contenido mismo no se incluyen en el registro referencial y pasan a otro registro junto con información adicional. La Tabla 2 muestra una síntesis de los metadatos utilizados.

Elementos de DCMI	Elementos de DC-Lib	Elementos propios de BDEAFIT
title, creator, subject, date, language, description, identifier, publisher, coverage, type, format, source, relation, contributor, rights	alternative, bibliographicCitation, abstract, audience, extent, edition, medium, location, tableOfContents	library, id, topografico, volumen, number, year, magazine, idarticulo.

Tabla 2: Relación de metadatos en BDEAFIT

A continuación se presentan la *Definición de Tipo de Documento* (DTD) tanto para un registro referencial como para un registro de contenido:

DTD para contenido referencial	DTD para contenido digital
<?xml version="1.0" encoding="UTF-8"?> *<! Entity definitions for rdfnsdecl, dcnsdecl, bdensdecl -->* *<!ELEMENT rdf:RDF (rdf:Description+)>* *<!ATTLIST rdf:RDF %rdfnsdecl; %dcnsdecl; %bdensdecl; >* *<!ENTITY % dcmes "dc:title | dc:creator | dc:subject | dc:description | dc:publisher | dc:contributor | dc:date | dc:type | dc:format | dc:identifier | dc:source | dc:language | dc:relation | dc:coverage | dc:rights" >* *<!ENTITY % bdemes "bde:library | bde:id | bde:topografico | bde:volumenes | bde:number | bde:cd | bde:year | bde:magazine | bde:idarticulo" >* *<!ELEMENT rdf:Description (%dcmes; | %bdemes;)* >* *<!—Dublin Core elements -->* *<!ELEMENT dc:creator (#PCDATA)>* *<!ATTLIST dc:creator id CDATA #REQUIRED>* *<!ELEMENT dc:subject (#PCDATA)>* *<!ATTLIST dc:subject id CDATA #REQUIRED>* *<!ELEMENT dc:identifier (#PCDATA)>* *<!ATTLIST dc:identifier* *type (topo | isbn | issn | urn) #IMPLIED>* *<!—similar definitions for otherDC elements -->* *<!—bde elements -->* *<!ELEMENT bde:cd (#PCDATA)>* *<!ELEMENT bde:id (#PCDATA)>* *<!ELEMENT bde:idarticulo (#PCDATA)>* *<!ELEMENT bde:library (#PCDATA)>* *<!ELEMENT bde:magazine (#PCDATA)>* *<!ELEMENT bde:number (#PCDATA)>* *<!ELEMENT bde:topografico (#PCDATA)>* *<!ELEMENT bde:volumenes (#PCDATA)>* *<!ELEMENT bde:year (#PCDATA)>*	*<?xml version="1.0" encoding="UTF-8"?>* *<!—other definitions -->* *<!ELEMENT rdf:Description (bde:id, dc:type, bde:content+)>* *<!ELEMENT bde:id (#PCDATA)>* *<!ELEMENT dc:type (#PCDATA)>* *<!ELEMENT bde:content (dc:description, dc:format, bde:file)>* *<!ELEMENT dc:description (#PCDATA)>* *<!ATTLIST dc:description type CDATA #REQUIRED>* *<!ELEMENT dc:format (#PCDATA)>* *<!ELEMENT bde:file (#PCDATA)>*

Tabla 3: DTDs para metadatos en BDEAFIT

Como se puede observar en los DTDs descritos en la tabla 3, el modelo de metadatos de BDEAFIT ha realizado algunas extensiones y modificaciones al DCMI, dichos cambios fueron incorporadas por los siguientes requerimientos:

1. Mejorar la velocidad de búsqueda en el sistema. Esto se logra mediante la adición de atributos a algunos elementos de DC, concretamente a *creator* y *subject*.
2. Se refinó el elemento *identifier*, declarando el atributo *type*, el cual mediante un vocabulario controlado, permite especificar el tipo de identificador al que hace referencia el contenido del elemento.
3. Información adicional en el dominio particular de las bibliotecas digitales y los sistemas de gestión bibliotecarios. Algunos fueron tomados del perfile de aplicación DC-Lib.

3.2 Carga de catálogos referenciales

Este módulo permite cargar o importar el catálogo referencial del *Sistema de Información Bibliotecario de la Universidad EAFIT* llamado SINBAD en BDEAFIT. Este proceso se realiza en procesamiento por lotes (batch),

mediante la extracción de la base de datos del SIB de los datos del catálogo (DB2TXT). Los datos del catálogo son generados en texto, de acuerdo a una convención definida por BDEAFIT para la carga de catálogos referenciales vía archivos de texto. Estos archivos son procesados a través del módulo de gestión de BDEAFIT, el cual permite importar un archivo texto del catálogo en BDEAFIT. El procesamiento realizado por el módulo de gestión consiste en dos etapas. La primera etapa procesa el archivo texto y genera con esta información documentos XML de acuerdo al modelo de metadatos RDF/DCMI (TXT2XML). La segunda etapa carga estos documentos XML en la base de datos XML (UPLOAD). Una vez los datos han sido cargados en BDEAFIT pueden ser utilizados por el módulo de búsqueda y recuperación de información así como otros procesos de gestión de contenido digital.

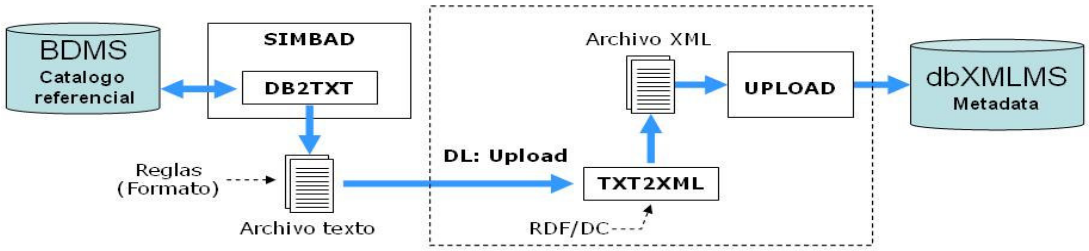


Figure 3. Obtención de información referencial o catálogo

El módulo DB2TXT es un componente de software que es implementado por el SIB posiblemente de manera intrusiva, esto eso, el módulo requiere entrar directamente a las bases de datos donde se encuentra almacenado el catálogo referencial y de acuerdo a unas reglas de formato, generar los archivos texto. La convención para generar los archivos texto así como el conjunto de registros se describe a continuación: Se recomienda un esquema de nombramiento para archivos de la forma "biblioteca_colección.txt", en donde biblioteca es el nombre de la biblioteca a la que pertenece y colección es el nombre de la colección a la cual pertenecen los registros. Aunque se puede incluir dentro de un mismo archivo todas las colecciones, una colección o un subconjunto de colecciones, se recomienda un archivo por colección para efectos de actualización incremental. (ej: bdeafit_libros.txt, bdeafit_video.txt). Un registro general contiene el siguiente formato:

"*titulo~~autores~~temas~~descripción~~editorial~~fecha~~colección~~id~~*
Topografico~~url~~idioma~~ubicación~~formato_digital~~derechos_autor~~otros_campos~~otros_campos"

El archivo texto utiliza una línea por registro y cada registro contiene dos separadores, uno para elemento (~~) y otro para repeticiones dentro de un elemento (%%). Las colecciones válidas son: "LIBRO", "TESIS", "PROYECTO", "REVISTA", "ARTICULO", "VIDEO", "AUDIO". Dependiendo del tipo de colección a que haga referencia, se interpreta cada registro de la siguiente manera:

Libros, tesis y proyectos de grado	*id – isbn* *otros_campos* - ninguno
Revistas y artículos	*id – issn* *otros_campos - revista~~volumen~~numero~~año*
Video y Audio	*id – issn o isbn* *otros_campos – id_medio~~duración*

En caso de no presentarse algún campo porque no tiene la información o porque para el tipo de colección no aplica, el campo debe estar vacío. Por ejemplo, si para la colección REVISTA, no existe el campo ISSN (porque no toda revista lo tiene) ni URL, el registro contendría la información así:

"*…~~REVISTA~~~~CB021~~~~ESPAÑOL~~…*".

El módulo TXT2XML toma como entrada los archivos texto generados por DB2XML y formateados según unas reglas y genera documentos XML en archivos, de acuerdo al formato de metadatos. A medida que se van generando los documentos XML, se van cargando a la base de datos XML a través del módulo UPLOAD. Ambos procesos son activados por el *administrador* o *publicador* de BDEAFIT a través del módulo de gestión.

3.3. Adición de Contenido Digital

Este módulo permite asociar y almacenar contenido digital al catálogo referencial en BDEAFIT, es precisamente a través de este módulo que se observa la transición de la biblioteca tradicional a la digital, ya que normalmente el catálogo referencial solo tiene relación con objetos físicos localizados en una biblioteca. Este módulo genera nuevos metadatos relacionados con el contenido digital y realiza modificaciones a los metadatos referenciales. Para almacenar los metadatos del contenido digital, se propone crear una nueva colección llamada "bdigital" dentro del repositorio de metadatos, de esta forma se haría referencia a la colección "bdeafit/bdigital" para almacenar dicho contenido. Se hará referencia al repositorio de metadatos referenciales como "bdeafit/catalogo".
El contenido digital es almacenado en servidores de datos, que dependiendo del tipo de objeto sería un servidor web para el material digital tradicional y un servidor multimedia para el contenido de video y audio.

Figure 4. Carga de la información referencial o catálogo.

Realmente este módulo no actualiza los metadatos referenciales, el proceso de actualización se realiza en la carga del catálogo referencial en el cual por cada registro que se vaya procesando, se verifica si tiene contenido digital asociado, en caso de encontrarlo en la base de datos relacional *bdigital_mysql*, actualiza el elemento DC relativo al identificador con un URL como contenido.
Básicamente hay dos grandes procesos que resumen la adición de contenido digital: (1) el proceso de gestión del contenido digital y (2) el proceso de publicación del contenido digital para que los módulos de búsqueda y recuperación accedan a ellos.
El proceso de gestión del contenido digital se encarga de crear, modificar o retirar metadatos relacionados con el contenido digital así como la carga o borrado de datos en los servidores respectivos. Cada uno de los materiales digitales es almacenado en servidores diferentes dependiente del tipo de objeto (tradicional o multimedia).
La información almacenada en la base de datos relacional (mysql) se considera un repositorio temporal el cual facilita la manipulación de la información de metadatos del contenido digital. Una vez la información esta estable, el publicador del sistema decide general o publicar los metadatos de contenido digital en el repositorio bdeafit/bdigital, como consecuencia de esto, el módulo de búsqueda y recuperación de información ya podrá acceder a dicha información. Este proceso de publicación consiste en transformar los datos almacenados en mysql en documentos XML bajo DC y luego ser almacenado en el repositorio de metadatos. Los datos mismos no requieren publicación ya que son almacenados en los servidores definitivos de la biblioteca digital. El proceso general de este módulo se resume en los siguientes procesos:

Consulta en el repositorio de metadatos referenciales: el usuario inicialmente debe localizar el registro de metadatos referencial sobre el que desea trabajar, para ello se desarrolló un módulo de consulta que permite por algunos criterios (titulo, autor, colección y topográfico) localizar el registro sobre el que realizará la asociación de contenido. Esta consulta se realiza sobre el repositorio de metadatos "bdeafit/catalogo". El resultado de esta consulta será un conjunto de registro, de los cuales el usuario selecciona uno con el cual trabajará.

Selección del registro referencial: Una vez se selecciona el registro de interés, el sistema recupera toda la información referencial y digital que contiene y el usuario entra en una fase de revisión del contenido digital (asociación/carga, modificación, borrado o visualización).

Asignación del material digital al registro referencial: De forma iterativa, se puede relacionar uno o más objetos digitales, para cada uno de ellos especificando el formato de la información, una breve descripción y la clase de documento que representa, por ejemplo capitulo, imagen, índice, portada, tabla de contenido, texto completo, audio,

video, etc. Finalmente se realiza un proceso de carga o transferencia de los archivos en los respectivos servidores de datos (web o multimedia).

En el servidor de datos tradicionales se almacenan los objetos digitales (pdf, html, imágenes, etc) así como un archivo de metadatos en XML. Este archivo es enlazado desde los metadatos referenciales. Los objetos digitales se almacenan clasificados por colección y topográfico. En las tablas 4 y 5 se presenta un ejemplo de la relación del contenido con los metadatos referenciales y digitales.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF>  <rdf:Description>
  <bde:id>1233323</bde:id>
  <dc:title>Modelo de simulación de redes</dc:title>
  <dc:creator id="G103">Juan Pérez</dc:creator>
  <dc:subject id="N201">Telecomunicaciones</dc:subject>
  <dc:identifier type="isbn">1233323</dc:identifier>
  <bde:library>bdeafit</bde:library>
  <bde:url>http://content.eafit.edu.co/bdeafit/bdigital/TESIS/1233323/index.xml</bde:url>
…
```

Tabla 4: Ejemplo de documento XML de un metadato referencial

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF> <rdf:Description>
<bde:id>1233323</bde:id>
<bde:content>
<dc:type>cover</dc:type>
<dc:description>Portada del libro</description>
<dc:format>image/gif</format>
<bde:file>http://content.eafit.edu.co/bdeafit/bdigital/TESIS/1233323/cover.gif</bde:file>
</bde:content>
<bde:content>
<dc:type>chapter</dc:type>
<dc:description>Capitulo 1: Introducción </description>
<dc:format>application/pdf</format>
<bde:file> http://content.eafit.edu.co/bdeafit/bdigital/TESIS/1233323/chapter1.pdf</bde:file>
</bde:content>
<bde:content>
<dc:type>chapter</dc:type>
<dc:description>Capitulo 2: Desarrollo de la obra</description>
<dc:format>application/pdf</format>
<bde:file> http://content.eafit.edu.co/bdeafit/bdigital/TESIS/1233323/chapter2.pdf</bde:file>
</bde:content>
<!-- otro contenido adicional -->
</rdf:Description></rdf:RDF>
```

Tabla 5: Ejemplo de documento XML de un metadatocon contenido digital

Publicación de metadatos del contenido digital: La publicación de los metadatos del contenido digital hace referencia al proceso de transformar los datos almacenados en la base de datos relacional (MySQL) en documentos XML, los cuales van a ser transferidos al servidor de contenidos y almacenados en el repositorio de metadatos de contenido (bdeafit/bdigital) para su indexación y acceso posterior por parte del módulo de consulta (DL:Search).

3.4. Creación de Contenido Digital

Este módulo permite crear objetos digitales nuevos en la biblioteca digital, esto es, permite crear los metadatos referenciales, de contenido y cargar los objetos digitales en los servidores respectivos. Como paso inicial, permite al administrador crear los metadatos descriptivos o referenciales, en el paso siguiente permite de una manera interactiva e iterativa crear los metadatos de contenido y realizar la transferencia de los archivos a los servidores de contenido. Toda la parte de gestión (creación, modificación, retiro) de contenido nuevo en la BD, se realiza en una base de datos relacional (mysql). El administrador puede publicar los metadatos en los repositorios referenciales y de contenido bajo solicitud. Este proceso transforma los datos almacenados en la base de datos relacional en documentos XML que son enviados tanto a los servidores de contenido como a los repositorios de metadatos XML. La figura 5 muestra el flujo general de trabajo en este módulo.

Inicialmente, solo los usuarios con roles de *administrador* o *publicador* pueden acceder a este módulo, como una mejora a este modulo se permitirá que otros tipos de usuarios puedan ingresar contenido modelando un servicio de gestión de documentos (*Content Management System*) ampliando usuarios con roles de *creadores* y *editores*.

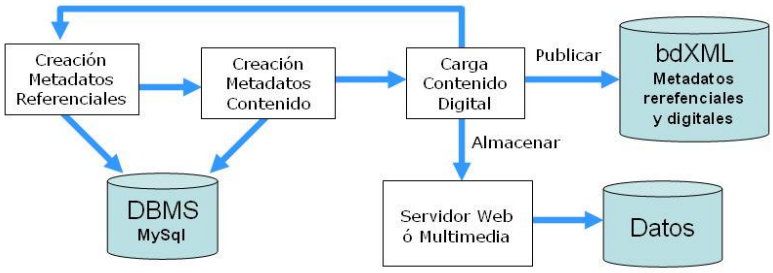


Figure 5. Creación de información.

3.5. Búsqueda y Recuperación de Información

El módulo de búsqueda y recuperación de información (DL:Search) permite a los usuarios del sistema localizar cualquier material referencial o digital que BDEAFIT contenga, si este contiene material digital, le permite descargarlo o reproducirlo en caso de ser material tradicional o multimedia respectivamente. El material mismo puede contener control de acceso, permitiendo especificar en el momento de carga del contenido digital, si este es público, válido dentro del dominio de EAFIT o protegido con una contraseña.

Este módulo realiza las búsquedas en los dos repositorios de metadatos (referencial y digital) y permite realizar consultas simples o avanzadas. En la búsqueda simple, el usuario digita cualquier texto y el sistema realiza una búsqueda general por todos los campos definidos para la búsqueda simple (titulo, autor y descripción) y en todas las colecciones contengan o no información digital.

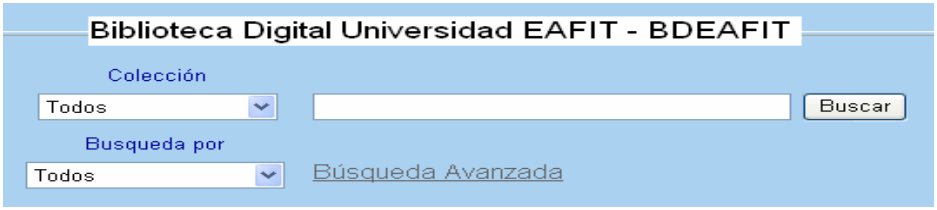


Figura 6. Búsqueda simple (DL:Search).

En la búsqueda avanzada, el usuario puede especificar algunos de los siguientes criterios de búsqueda las cuales pueden ser combinadas mediante operadores lógicos "Y" o "O":

- Por colección
- Por tipo de documento
- Por título
- Por autor
- Por tema
- Contenido digital

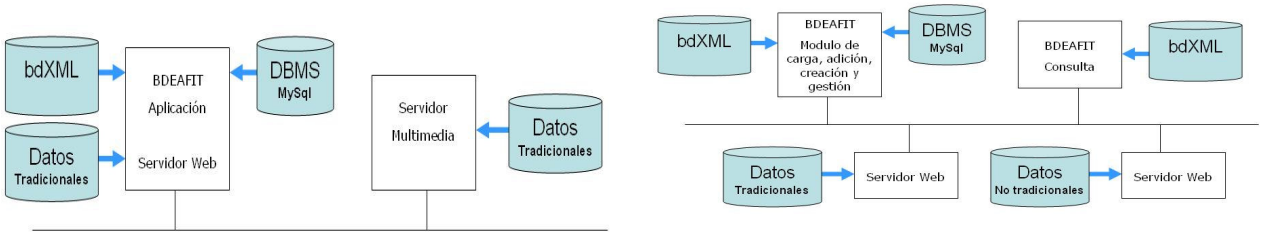
Figura 7. Búsqueda avanzada (DL:Search).

Los criterios de "colección" y "tipo de documento" son vocabularios controlados en la interfaz del usuario, los criterios "titulo", "autor" y "tema" son términos digitados por el usuario y pueden ser entrados parcialmente y utilizar meta-

caracteres o comodines, se puede especificar si los términos son exactos o parciales y finalmente especificar si desea buscar en todo el repositorio de metadatos o solo los que contienen información digital.
La búsqueda se realiza en la base de datos XML, en donde se encuentran almacenados los metadatos referenciales y digitales, y por ser una base de datos nativa en XML, se utiliza la especificación XPATH y XQUERY como lenguaje formal de consulta en la base de datos. Estas especificaciones están optimizadas para realizar consultas sobre conjuntos de documentos XML.
Una vez el usuario ha obtenido el resultado de la consulta, éste puede navegar a través de los resultados y cuando encuentra un registro de interés, ingresa a él para conocer más detalles. En caso del registro contener información digital, aparecerá en la interfaz del usuario el URL para acceder a dicho contenido.

3.7. Implementación

Aunque a lo largo de este artículo se han detallado algunos aspectos de la implementación, a manera de síntesis describiremos el ambiente de implementación y despliegue de BDEAFIT. Está implementado en Java (J2SE 1.5), MySQL como base de datos relacional, eXist [11] como base de datos XML y API de desarrollo para todos los módulos de la aplicación. Servidor Web Apache para almacenar y recuperar el contenido tradicional y un servidor multimedia basado en Windows Media Services 9. Utiliza el marco de trabajo *struts* [12] [13] para los módulos de carga de catálogos, adición y creación de contenido digital. El módulo de consulta está realizado directamente sobre eXist basado en XPATH, XQUERY y XSLT para generar el contenido a HTML hacia navegadores tradicionales.
Actualmente se tienen alrededor de 200.000 registros referenciales indexados y unos 200 registros con contenido digital. Se tiene una concurrencia de 50 usuarios consultando con un tiempo de respuesta promedio de menos de 5 segundos, lo que es muy aceptable. Está desplegado en dos servidores físicos (figura 8(a)), uno en donde se encuentra toda la aplicación de BDEAFIT con todos sus módulos además del servidor Web para almacenamiento de objetos digitales tradicionales y otro servidor para el contenido multimedia para el servicio de Video/Audio bajo Demanda. Sin embargo, la arquitectura ideal de despliegue de la aplicación está dado por la figura 8(b) en la cual se cuenta con cuatro servidores



(a)Arquitectura actual de despliegue. *(b)Arquitectura futura de despliegue.*

Figura 8. Arquitecturas de despliegue

La mayoría del software y plataformas utilizadas por BDEAFIT es abierta, estándar y de libre distribución, salvo el servidor multimedia, aunque en otros tipos de implementaciones se podrían utilizar otros servidores multimedia o inclusive ser almacenados en servidores Web.
El proceso de desarrollo de BDEAFIT se ha realizado mediante un esquema incremental, iniciando con una primera fase en la cual solo se contenía información referencial y adición de contenido, una segunda fase en la cual se podían adicionar objetos nuevos (ya sea de procesos de digitalización o de objetos digitales nativos) y la fase actual en la cual se integra con otras bibliotecas digitales (por ejemplo BDEAFIT y BDEI).

4. TRABAJOS FUTUROS

BDNG y sus aplicaciones derivadas como BDEAFIT, BDEI y Metabiblioteca están en constante evolución, incorporando nuevas funcionalidades e investigaciones tanto generadas desde el mismo grupo de investigación a través de proyectos, tesis de maestría y proyectos fin de carrera así como los adelantos científicos en esta comunidad de aplicación. Como aspectos generales de la arquitectura BDNG se están abriendo líneas especificas de acción las cuales se resumen en la siguiente lista:

- Incorporar mecanismos de búsquedas avanzadas.
- Implementación de BDNG como un *framework* de software que facilite el desarrollo de aplicaciones.
- Incorporar un mecanismo de Manejador de Contenido, que permita modelar todos los roles de estos servicios (creador, editor, publicador y administrador).

- Incorporar mecanismos de protección de información y propiedad intelectual.
- Integrar diversas fuentes de contenidos representados en producción propia, procesos de digitalización, integración de las bases de datos internacionales referenciales y de texto completo, etc. con una vista dada por la BD.

Búsquedas Avanzadas: El objetivo es diseñar un modelo de búsqueda avanzada que mejore la recuperación de información digital a los usuarios. Se pretende incorporar aspectos como búsquedas multilenguaje, para lograr acceso a la información tanto en español como en otros idiomas. Se ha planeado agregar otro tipo de metadatos como son los sinónimos en otros idiomas (Multi-lenguaje), tesauros/ontologías y los propios de un análisis de frecuencia (ranking). La expansión del significado o dominio de la consulta se hará usando diccionarios [14]. Además se almacenará en una base de datos de conocimiento, aspectos relacionados con los usuarios como son las preferencias de consulta, y las ultima consulta realizada. Incluir variantes morfológicas (género y número) para los términos de una consulta. También será posible realizar un análisis de tipo lexicográfico, descomponiendo la consulta en sintagmas nominales, preposicionales y verbales con el fin de resolver la ambigüedad del proceso. Este proyecto actualmente está siendo desarrollado como una tesis de Maestría en Ingeniería Informática en la Universidad EAFIT.

BDNG como un *framework* software: La versión actual requiere que la aplicación concreta sobre BDNG realice modificaciones e implementaciones sobre la implementación software de BDNG, la idea es desarrollar el software de BDNG a modo de *framework* que permita desplegar fácilmente aplicaciones sobre ésta sin requerir modificaciones o ampliaciones al código fuente.

Manejador de contenidos: Ampliar los roles de creación y carga de contenido digital en la BD, de tal manera que se pueda asociar flujos de trabajo en la producción de contenido digital. Esto se logra mediante la formalización de los roles de usuario creador y editor, los cuales actualmente no están implementados [15]. Como un caso concreto, se ha propuesto implementar un flujo de trabajo que facilite el proceso de creación de la producción académica y científica de la Universidad EAFIT representada en las tesis de maestría, proyectos fin de carrera, artículos, reportes de investigación, entre otros.

5. CONCLUSIONES

La propuesta de diseño e implementación de bibliotecas digitales basadas en la arquitectura BDNG como es el proyecto BDEAFIT, ha demostrado la efectividad en la evolución de las bibliotecas tradicionales en digitales, en las cuales el usuario mediante una única interfaz, consulta el catálogo referencial del antiguo sistema bibliotecario de la universidad (SINBAD) y puede recuperar el contenido digital almacenado en sus propios repositorios de datos o distribuidos a través de Internet en otras bibliotecas digitales. Estas bibliotecas digitales a diferencia de Internet, plantean mecanismos de acceso organizado y estructurado mediante modelos eficientes de metadatos que permiten al usuario recuperar los datos que realmente está buscando. Internet, a pesar que es un gran universo de contenido, adolece de mecanismos eficientes de organización y clasificación de la información, los cuales muchas veces son accedidos a través de los bien conocidos *motores de búsqueda* (*google*, *yahoo*, etc). Si bien se están realizando importantes avances en desarrollar una Web semántica, actualmente el concepto de bibliotecas digitales resulta ser el mecanismo mejor estructurado para acceso organizado a la información.

BDNG fue diseñado desde el comienzo como un marco de trabajo general que pudiese ser utilizada en muchos proyectos y campos de aplicación. Actualmente se han desarrollado tres proyectos importantes sobre esta arquitectura: la biblioteca digital de la Universidad EAFIT, la biblioteca digital de EAFIT Interactiva, nuestro sistema de *e-learning* y una iniciativa para la integración de bibliotecas digitales y referenciales que cuenta con más de 10 instituciones de educación superior integrando sus catálogos y contenido digital a través de nuestro sistema.

BDNG presenta un sistema abierto y estándar, que utiliza en su mayoría componentes de software abierto y de libre distribución. Utiliza ampliamente bases de datos XML nativas para el almacenamiento e indexación de metadatos, lo que deriva en nuevas formas de búsqueda y recuperación diferentes a las clásicas basadas en bases de datos relacionales. A nivel de metadatos los implementa bajo RDF y Dublín Core, con algunas extensiones y codificado en XML. A nivel de contenido no solo maneja datos tradicionales sino que ha considerado desde el comienzo la incorporación de datos multimedia.

Finalmente, las bibliotecas digitales podrían ayudar a nuestros países en vías de desarrollo a reducir la brecha digital y a facilitar los procesos de inclusión digital tanto entre los países más desarrollados y nosotros como dentro de nuestras mismas sociedades. El gran beneficio de una biblioteca digital en nuestros entornos no es únicamente poder acceder a

fuentes universales de conocimiento, sino fomentar la producción y digitalización de nuestro propio contenido hacia el mundo. Más información acerca del proyecto en: http://dis.eafit.edu.co/projects/bdng

REFERENCIAS

[1] Alberto Laender, Marcos André Gonçalves, Pablo Roberto. BDBComp: Building a Digital Library for the Brazilian Computer Science Community. JCDL 2004, Tuscon, AZ, USA.. Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries (JCDL'04), ISBN 1-58113-832-6/04. http://csdl.computer.org/comp/proceedings/jcdl/2004/2493/00/24930023.pdf.

[2] IP2, Digital Libraries. Future Directions for a European Research Programme. DLIP2 - International Projects. DELOS Brainstroming Report, San Cassiano, Alta Badia--Italy, June 13-15, 2001. DELOS-ERCIM, 2001.33 p. http://www.dli2.nsf.gov/internationalprojects/eu_future.html#_Toc53339570.

[3] Chau, M. e tal. Comparison of Two Approaches to Building a Vertical Search Tool: A Case Study in the Nanotechnology Domain. Proceedings of The Second ACM/IEEE-CS Joint Conference on Digital Libraries, julio de 2002, p:135-144 (Última vez consultada en marzo 10 de 2004). http://ai.bpa.arizona.edu/~mchau/papers/VerticalSearchTool.pdf.

[4] Borbina, J, N. Freire and J. Neves. BND: The Architecture of a National Digital Libray, ACM/IEEE Joint Conference on JCDL'04, pp: 21-22. June 2004.

[5] Carl Lagoze, Herbert Van de Sompel. The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). Version 2.0 of 2002/06/14, http://www.openarchives.org/OAI/openarchivesprotocol.html

[6] WWW Consortium, Resource Description Framework (RDF), http://www.w3.org/RDF.

[7] DCMI. Dublin Core Metadata Initiative Overview. Metadata Associated and OCLC Research, 2004. http://www.dublincore.org/.

[8] E. Méndez. Metadatos y recuperación de información: Estándares, problemas y aplicabilidad en bibliotecas digitales. Ediciones Trea, 2002. 429 p. Mención de serie Biblioteconomia y administración cultural no. 66. ISBN: 84-9704-055-4.

[9] Rebecca Guenther. DC-LIB, Library Application Profile 2001-10-12. DCMI-Libraries Working Group. http://dublincore.org/documents/2001/10/12/library-application-profile

[10] Alejandro Bia, Irene Garrigós, Jaime Gomez: Personalizing Digital Libraries at Design Time: The Miguel de Cervantes Digital Library Case Study. ICWE 2004: 225-229. ISBN: 3-540-22511-0.

[11] A. Chaudri, et al. XML Dates Management: Native XML and XML-Enabled Database Systems. ISBN: 0-201-84452-4. Published by Addison Wesley Professional, March, 2003.

[12] Daniel Fernández Lnvin. Definición de una Arquitectura de Software para el Diseño de Aplicaciones WEB Basadas en Tecnología Java-J2EE, Universidad de Oviedo. http://di002.edv.uniovi.es/~dflanvin/doctorado/ArquitecturaJ2EE.PDF.

[13] The Apache Software Foundation. Framework for building Java web applications. 2005. http://struts.apache.org/

[14] M. Fernandez, et al. DelfosnetX: A Workbench for XML-Based Information Retrieval Systems. Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000. IEEE Computer Society. Los Alamitos, CA. September. ISBN 0-7695-0746-8, pp. 87-85. http://csdl.computer.org/comp/proceedings/spire/2000/0746/00/0746toc.htm

[15] MacKenzie Smith. DSpace: An Open Source Institutional Repository for Digital Material. D-Lib Magazine, October 2002. Volume 8 Number 10. ISSN 1082-9873.

EXPERIENCIAS EN EL DESARROLLO DE UNA PLATAFORMA DE SERVICIOS DE E-COMMERCE PARA UNA COMUNIDAD VIRTUAL DE NEGOCIOS

Martha Eliana Mendoza
Universidad del Cauca, Departamento de Sistemas,
Popayán, Colombia
mmendoza@unicauca.edu.co

Luz Marina Sierra
Universidad del Cauca, Departamento de Sistemas,
Popayán, Colombia
sierra@unicauca.edu.co

Jorge Jair Moreno
Universidad del Cauca, Departamento de Sistemas,
Popayán, Colombia
jjmoreno@unicauca.edu.co

Roberto Carlos Naranjo
Universidad del Cauca, Departamento de Sistemas,
Popayán, Colombia
rnaranjo@unicauca.edu.co

Abstract

This article describes the current problem in the business environment from the Cauca region – Colombia (South America), and the proposed solution called Project CVN (Spanish initials) "Business Virtual Community - for the Cauca region - Internet Commercial Platform or BVC". Based on a markets research, the architecture of the added value services conceived by the project is described; these services support advertising, collaboration, B2C, and B2B activities framed within the virtual environment of the community. Below, the business model proposed for the community is explained on two major áreas: the services portolio and the sustainability strategies for the community. Lastly the experiences and the learnt lessons throughout the implementation of the project are exposed.

Keywords: E- Communities, E-Commerce, E-Business Models, Marketing Strategies, Usability and Sociability, E-Commerce on Developing Countries.

Resumen

Este artículo describe el problema actual en lo relativo al entorno de negocios para la región del Cauca – Colombia (América del Sur) y la solución propuesta llamada "Comunidad Virtual de negocios para el Cauca – Plataforma comercial en Internet – CVN". Basado en una investigación de mercados, la arquitectura de servicios valor agregado concebida para el proyecto es descrita; estos servicios soportan actividades relacionadas con la publicidad, colaboración, B2C y B2B todos enmarcados dentro del ambiente virtual de la comunidad. Más adelante se explica el modelo de negocios propuesto para la comunidad en dos grandes áreas: el portafolio de servicios y las estrategias de sostenibilidad para la comunidad. Finalmente se exponen las experiencias y lecciones aprendidas durante la implementación del proyecto.

Palabras claves: Comunidades Virtuales, E-Commerce, Modelos de E-Business, Estrategias de Marketing, Usabilidad y Sociabilidad, Comercio electrónico en países en vía de desarrollo.

1. INTRODUCCION

Desde el nacimiento hasta la muerte, damos forma y somos formados por las comunidades a las que pertenecemos. Para bien o para mal esas comunidades influencian nuestro lenguaje, como pasamos el tiempo, lo que consideramos importante, con quienes interactuamos y que naturaleza tiene esa interacción. Algunas comunidades crean condiciones favorables para interacciones sociales fuertes, mientras que otras son todo lo contrario, algunas son constructivas y algunas no[6]. Desarrollar una comunidad virtual es una actividad compleja, puesto que el concepto de las comunidades virtuales tiene diversos significados para diferentes personas[6]. Sin embargo, una comunidad virtual presenta cuatro aspectos fundamentales que son: **personas**, **propósito, políticas** y los **sistemas de software**.

Dentro de este marco conceptual de comunidad virtual, se ha desarrollado CVN, la cual constituye una iniciativa de cooperación entre la Universidad del Cauca[1], la Empresa privada (Cygnus Tecnologías[2]) y COLCIENCIAS[3], que adecua las TI a las necesidades comerciales específicas del departamento del Cauca-Colombia. Dentro de los problemas que presenta actualmente las empresas del Departamento del Cauca, se destacan tres aspectos principales: ausencia de asociatividad-colaboración entre estas, distanciamiento cliente/empresa por causa del deterioro de los canales de comunicación y desconocimiento-temor-desconfianza de las empresas respecto a las ventajas del uso de TI[3]. En Colombia existen iniciativas similares tales como: Plaza Orbital[4] la cuál busca impulsar productos y servicios de la ciudad de Medellín, De Remate[5] líder latinoamericano en ventas de productos en subastas y TAMPU[6] enfocado a impulsar la imagen turística de Popayán y el Cauca, entre otros. Frente a las iniciativas anteriores, CVN surge como una alternativa de solución de los problemas presentados, planteando un concepto de hacer negocios que congrega las empresas de la región, mediante un ambiente virtual que ofrece servicios de valor agregado orientados a la publicidad, el B2C y B2B. La CVN, favorecerá condiciones culturales y tecnológicas que motiven la asociatividad y colaboración entre empresas, así como la formulación de estrategias de promoción de productos y servicios que favorezcan el acercamiento eficaz de estas con sus clientes con el propósito de fortalecer la confianza en las TI como potenciadoras de la competitividad y productividad empresarial.

Este artículo pretende mostrar los resultados del proyecto de investigación CVN, en el ítem 2, se presenta brevemente las etapas que se siguieron para el desarrollo del proyecto, se describe la investigación de mercados que se realizo y las conclusiones de la misma, se explica el modelo de negocios propuesto para la comunidad en dos grandes áreas: el portafolio de servicios y las estrategias de sostenibilidad para la comunidad. Luego se presenta los servicios de valor agregado (que soportan actividades relacionadas con publicidad, colaboración, B2C y B2B) que ofrece el proyecto y su arquitectura, se explican las estrategias de usabilidad y sociabilidad para la CVN y se presenta como va la puesta en marcha de la CVN. Para terminar, en el ítem 3, se exponen las experiencias obtenidas durante la investigación y las conclusiones que arroja la implementación del proyecto.

2. EVOLUCIÓN Y DESARROLLO DEL PROYECTO

La realización del proyecto CVN se desarrollo en las siguientes etapas:

1. Elaboración de una Investigación de Mercados en la región del Cauca.
2. Concepción del Modelo de Negocios el cual comprende:
   - Concepción y diseño del Modelo de servicios o portafolio.
   - Concepción y diseño de las estrategias de sostenibilidad.
3. Ambientes Virtuales para la plataforma de servicios de CVN.
4. Especificación de las estrategias de Usabilidad & Sociabilidad para la CVN.
5. Estrategia de transferencia y puesta en marcha de CVN bajo la marca Netgociemos.com.

2.1 Investigación de Mercados

Propósito. La investigación de mercados[4] realizada buscó, en primer lugar, identificar-cuantificar la oferta/demanda regional y nacional sobre productos-servicios soportados en TI a través de CVN. En segundo lugar, determinar perfiles y tendencias empresariales, que permitieran identificar oportunidades y amenazas del entorno con el propósito de diseñar estrategias de comercialización y sensibilización de las transacciones vía Internet. Se formularon las siguientes hipótesis de investigación:

---

[1] Universidad del Cauca. Portal Institucional en Internet. http://www.unicauca.edu.co y Grupo de Investigación en Tecnologías de la Información – Unicauca. http://atenea.unicauca.edu.co/~gti/

[2] CYGNUS Tecnologías. Website Corporativo. http://www.cygnus.com.co.

[3] COLCIENCIAS. Portal Gubernamental de Investigación en Internet. http://www.colciencias.gov.co

[4] Plaza Orbital Portal Comercial en Internet. http://www.plazaorbital.com.co

[5] De Remate Portal Comercial en Internet. http://www.deremate.com

[6] TAMPU Portal Institucional en Internet. http://www.tampu.unicauca.edu.co

1. ¿Existe una demanda insatisfecha de servicios Internet y comercio electrónico?
2. ¿Existe un desconocimiento del negocio, oportunidades y riesgos derivados de la comercialización por Internet?
3. ¿Existen barreras técnicas-económicas en la región, que se oponen a la comercialización por Internet?
4. ¿Las ofertas de soluciones integrales de Internet y comercio electrónico en la región son inexistentes?

Ejecución. Durante la ejecución de la investigación, los aspectos críticos de información identificados para validar las hipótesis anteriormente formuladas fueron:

1. Verificar la justificación del proyecto mediante la existencia de una demanda significativa para una solución como CVN, por parte del sector productivo de la región.
2. Identificar las amenazas y oportunidades del proyecto a partir de un estudio del uso actual de las IT en la región.
3. Finalmente estimar la oferta de productos-servicios en el Cauca con posibilidades de comercialización a nivel nacional o internacional.

Ficha Técnica y Muestra. El universo de la muestra, lo conformaron empresas y asociaciones productivas del departamento del Cauca con cobertura departamental, nacional o internacional. La muestra consistió de 54 empresas de 12 áreas de negocios diferentes. El tipo de muestreo fue "No probabilístico" por conveniencia ya que el proceso de selección de las empresas fue cuidadosamente guiado por características específicas que le interesaban al proyecto. Este tipo de muestreo, redujo los costos durante el proceso de la obtención de la información, permitiendo que el investigador decida que elementos de la población pasan a formar parte de la muestra en función de la disponibilidad de los mismos (proximidad con el investigador, amistad, etc.). El muestreo fue adecuado para esta investigación, debido a la inexistencia de información confiable en los organismos de control, por ejemplo: en la Cámara de Comercio existen registros de miles de empresas con valores en activos que no concuerdan con la realidad y en varios casos las empresas están sólo en el papel. Estas condiciones hacen difícil estimar el universo, la población y la muestra, sumándole el hecho de que las empresas objeto de estudio deben ser del ámbito municipal, departamental, nacional o internacional, es decir, no aplican empresas cuyo ámbito sea un barrio un sector o una comuna. Las áreas fundamentales que se tuvieron en cuenta para la elaboración del instrumento de recolección de información fueron:

1. El acceso a Internet.
2. La experiencia comercial en Internet.
3. El uso y la frecuencia de las herramientas Internet.
4. Los perfiles de compra y venta a través de Internet.
5. El interés en adquirir herramientas que permitan la asociación a través de Internet.
6. Las condiciones de participación del sector empresarial en el proyecto.

Conclusiones. La investigación de mercados, se centró en aspectos relacionados con Internet, específicamente: Uso, experiencia, problemas, compras, ventas, asociatividad, publicidad, interés y condiciones de participación en el proyecto. Los resultados arrojaron condiciones favorables para la ejecución y sostenibilidad de la CVN, en cada uno de los aspectos arriba mencionados. Las conclusiones de la investigación, permitieron diseñar el portafolio de servicios y elicitar los requerimientos técnicos y funcionales de la CVN.

2.2 Concepción y diseño del Modelo de Negocios

El modelo de negocios de la CVN[5] se sustenta en dos partes fundamentales que son: un Modelo de servicios y unas Estrategias de sostenibilidad, que se explican brevemente en los ítems siguientes. El modelo y las estrategias permitirán la operacionalización de la CVN.

2.2.1Concepción y Diseño del Modelo de Servicios o Portafolio

El modelo de servicios concebido para la CVN, se distribuye en tres grandes áreas, a saber: servicios de publicidad y colaboración, servicios de B2C y servicios de B2B. Cada área, está soportada por un ambiente virtual basado en web que implementa los servicios que ofrecen valor agregado a los miembros de la comunidad De otra parte, los servicios de la CVN, se organizan en tres tipos de suscripciones (gratuita, estándar y plus), según sea su complejidad y costo se tiene una correspondencia entre los servicios a ofrecer y los tipos de suscripción. A continuación se describen brevemente el propósito de cada componente del modelo de servicios.

- **Componente de Publicidad y colaboración.** Apoya el acercamiento entre empresas y sus clientes mediante estrategias de impulso a la promoción de productos/servicios y colaboración-intercambio de información entre las empresas. Los principales servicios que ofrece son: catálogo de productos/servicios (oferta y demanda), motor de búsqueda, clasificados, información sobre eventos, administración de la información y charlas moderadas por expertos.

- **Componente de B2C.** Provee los siguientes servicios: carro de compras, perfiles personalizados de consumo, análisis de información de acuerdo a los hábitos de compra de los usuarios de la comunidad y subastas-bajastas entre usuarios de la comunidad.
- **Componente de B2B.** Ofrece el soporte para la cooperación entre empresas con el propósito de potenciar un mayor poder de negociación empresarial, la consecución de mejores precios mediante servicios de Cotización en línea, Agregación de la demanda/oferta de productos-servicios e información comercial de la ciudad.

2.2.2 Concepción y Diseño de las Estrategias de Sostenibilidad

Las estrategias de sostenibilidad definen el costo estimado sobre la prestación de los servicios de la empresa ***Netgociemos.com***[7], lo que permitirá garantizar a largo plazo la evolución y crecimiento de la comunidad. La investigación de mercados y las proyecciones realizadas permitieron concluir dos fuentes de ingreso para garantizar la sostenibilidad de la CVN, las cuales se describen a continuación:

1. Suscripciones o membresías. Orientadas a aquellas empresas o clientes cuyo interés es participar activamente en la comunidad y son:

- **Membresía Empresarial gratuita.** Tiene como propósito, permitir que los clientes utilicen los servicios básicos como: páginas amarillas, FrontPage empresarial, búsqueda sobre productos y servicios de la CVN, envío/recibo de boletines semanales, participación en subastas y bajastas, etc.
- **Membresía Empresarial Estándar**. Provee a las empresas de servicios de valor agregado tales como: acceso a charlas electrónicas orientadas por especialistas, recopilación de información de Evaluación que los Clientes realizan sobre las Empresas en aspectos relacionados con: productos y servicios, atención al cliente, cumplimiento y soporte post-venta, configuración de soluciones integrales de servicios para clientes etc.
- **Membresía Empresarial Plus.** Orientada a las empresas que deseen convertirse en miembros activos de la comunidad, con servicios adicionales como: soporte en línea, servicio de agregación de la demanda para proveedores, servicio de agregación de la oferta para grandes clientes, configuración de soluciones integrales de servicios para clientes, cotización de órdenes de compra Inter-empresariales, entre otros.

2. Venta de servicios de valor agregado. Estos están concebidos para aprovechar la información comercial, con el propósito de favorecer la competitividad empresarial, al mismo tiempo que potencian sus capacidades de negociación y comunicación. Los servicios contemplados son: Publicidad (Banners), Branding, Marketing Directo, Obtención de Datos, Captación de Clientes, Promociones, Retención de Clientes. Con miras a definir un portafolio de servicios adecuado a las empresas y clientes potenciales de ***Netgociemos.com*** y que a su vez garantice la sostenibilidad de la comunidad en su fase de producción se realizaron los siguientes estudios:

- Análisis de las cadenas productivas del Cauca, tomando como base la cadena piscícola para determinar las necesidades de información y las diferentes problemáticas presentadas al interior de ellas, concluyéndose que los servicios implementados por la CVN son una alternativa para apoyar la solución de las problemáticas planteadas como son: la dificultad en los procesos de comunicación lo que imposibilita el fortalecimiento organizacional, la falta de definición de procesos ágiles, la agremiación con miras a buscar nuevos mercados para productos competitivos y rentables.

- Revisión de modelos de negocios según los servicios a ofrecer, en el cual se determinó, que ***Netgociemos.com*** se presentaría como un portal que busca no sólo atraer clientes y sino también mantenerlos.

- Generación del Plan estratégico de Mercado, el cual define la introducción en el mercado y la puesta en marcha de la CVN, y del cual se obtuvo lo siguiente:
  - La Misión, Visión, Políticas, Principios, Propósitos que regirán a ***Netgociemos.com***.
  - Las condiciones económicas para la CVN son favorables, teniendo en cuenta el apoyo que se está ofreciendo para la creación de empresas y las expectativas que se tienen en los nuevos acuerdos económicos que se están impulsando.
  - La inversión del estado y apoyo a programas que permiten masificar el uso de las tecnologías de la información favorece la vinculación de clientes potenciales a la CVN.
  - Las ventajas ofrecidas por las TI, facilitan la comunicación de los empresarios con sus clientes, proveedores, socios, al igual que el acceso a información oportuna.

[7] Nombre del sitio Web de la CVN.

- En el análisis de las cinco fuerzas competitivas que plantea el "Diamante competitivo" de Porter[4], el poder esta concentrado en los productos sustitutos y las barreras de entrada, el nivel de integración de los proveedores esta medianamente consolidado, pero en el futuro tiende a ser alto.

Teniendo en cuenta los resultados obtenidos en los estudios y análisis anteriores y cruzándolo con los servicios implementados para la CVN, se determinó que para efectos del portafolio comercial que va a ofrecer a sus usuarios la CVN Netgociemos.com, los servicios descritos en el Numeral 2.2.1 se agrupan, de tal forma que permitan garantizar la sostenibilidad económica de la empresa en su fase de producción según se describe en la Tabla 1.

Este portafolio de servicios permite definir diferentes esquemas de tarifas dependiendo de las necesidades de los clientes potenciales, el precio de los servicios será diferenciado según el mercado al cual oriente ***Netgociemos.com*** (local, regional o Internacional), de tal forma que en el mercado local se ofrecerán servicios a muy bajo precio para favorecer la utilización del servicio; en el mercado regional y nacional el costo de los servicios redondeará un precio por debajo del promedio de la competencia como estrategia de penetración del mercado y para capturar la cuota mínima que garantice la sostenibilidad de la CVN. Después del primer año de actividad se redefinirá el precio de los servicios tomando como base el análisis portafolio según se clasifique la tendencia de los servicios.

2.3 Ambientes Virtuales para la plataforma de servicios de CVN.
CVN es una plataforma tecnológica de servicios que soporta operaciones entre empresas (B2B), clientes y empresas (B2C), clientes a clientes (C2C) y servicios de colaboración y publicidad (P&C). A continuación se describe la arquitectura y los servicios concebidos para la plataforma de CVN.

PORTAFOLIO DE SERVICIOS			Inscripción	Mensual	Anual	Comisión	Pago Único
TARJETA VIRTUAL B2C	Es una tarjeta de presentación virtual dirigida a profesionales independientes o empresarios que quieran publicitar su nombre o su marca con datos básicos como dirección, teléfono, correo electrónico.	Hosting con información de la empresa o servicio.	**X**	**X**	**X**		
VITRINA VIRTUAL B2C	Es una "ventana" donde los empresarios pueden publicitar sus productos en forma dinámica ya que permite a sus clientes interactuar con el catalogo para conocer detalles de un producto o servicio.	PAD[8]	**X**	**X**	**X**		
ENDA VIRTUAL B2C	Ofrece el servicio de vitrina virtual, sumado a la posibilidad de la compra en línea haciendo uso del carrito de compras.	PAD + Carrito de compras	**X**	**X**	**X**	**X**	
EMPRESA VIRTUAL B2B	Ofrece el servicio de vitrina virtual, adicionado a los servicios de asociatividad para empresas permitiendo a diferentes empresas atender una solicitud de compra (soluciones integrales) y sumar demandas para conseguir un mayor poder de negociación (agregación de la demanda)	PAD + Carrito de compras + Grupos de Compras y Agregación de la Oferta + Soluciones Integrales	**X**	**X**	**X**	**X**	
SERVICIOS PLEMENTARIOS B2C-B2B	Las subastas y bajastas son herramientas efectivas de promoción de productos y servicios. Puede ser adquirida por cualquier miembro de la comunidad que haya adquirido alguno de los cuatro servicios	Subastas / Bajastas		**X**		**X**	

[8] PAD incluye Servicios de publicidad, administración de productos, servicios de colaboración.

	básicos.						
PUBLICIDAD Y COLABORACION	Servicios orientados a las empresas o profesionales, que les ofrecen la posiblildad de promocionar, publicitar e interactuar con su mercado objetivo.	Baner Chat Foro					**X**

Tabla 1 Portafolio de Servicios de la CVN

2.3.1 Arquitectura Lógica de la plataforma de servicios para la CVN

El modelo arquitectónico elegido para desarrollar los ambientes virtuales de la comunidad consistió en el patrón **servicios-dominio-aplicación[2],** que facilita el diseño de las aplicaciones ya que las divide en capas fácilmente modelables independientemente y provee mecanismos de comunicación entre ellas a través de mensajes que permiten ejecutar cierta lógica del negocio. En la Figura 1, se muestra (parcialmente) la capa del dominio, en donde se visualizan las principales clases y la relación entre estas en la aplicación de carro de compras, catalogo de productos y suscripciones.

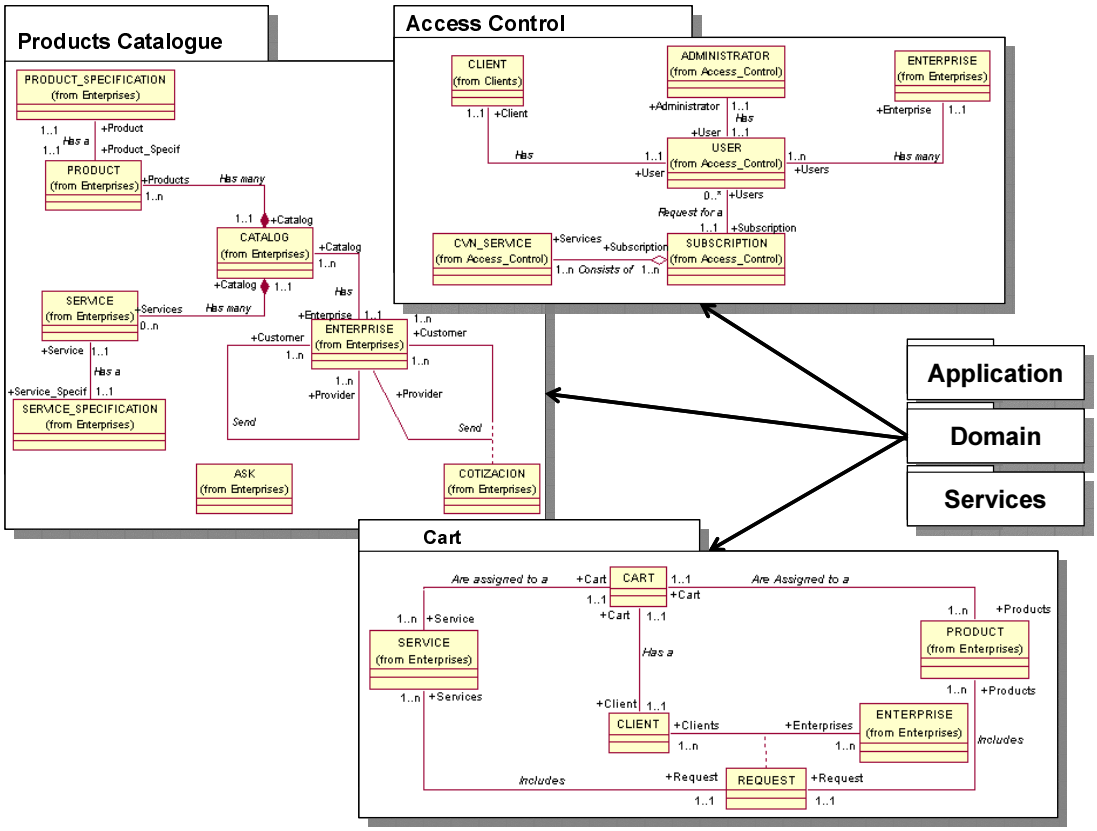


Figura 1: Arquitectura Lógica - Vista Parcial de la Capa de Dominio

2.3.2 Ambientes virtuales y servicios ofrecidos

- Servicios ofrecidos por el Ambiente de B2C.
  - **Carro de Compras**. Permite comprar productos de las empresas a través de solicitudes de pedido.
  - **Catalogo de productos**. Permite ver el inventario de productos de las empresas.
  - **Servicio de Inteligencia de negocios.** Ofrece estadísticas de ventas de la empresa, de acuerdo a las categorías de productos de la empresa. Análisis de Minería de datos para encontrar tendencias en las

ventas mediante técnicas de asociación, y así encontrar y aplicar estrategias de mercadeo. También administración de los pedidos, que permite determinar y procesar sus pedidos.
- **Servicio de C2C o de subastas – bajastas (**subastas inversas**)**. Publicación de subastas y bajastas, con el propósito de vender y comprar artículos. Participación y colocación de ofertas, el cual les permite hacer preguntas u ofertas por artículos en una subasta o bajasta.

- Servicios ofrecidos por el Ambiente de B2B
  - **Soluciones integrales**: Permite a la CVN ofrecer a las empresas la posibilidad de adquirir todos los productos requeridos en un solo ciclo de compra. Adicionalmente se ofrece un agente inteligente que notifica rápida y efectivamente, sobre las solicitudes que los clientes han enviado.
  - **Grupos de compra**: Para que las empresas de la CVN puedan conseguir beneficios comunes como la disminución de los precios de compra y obtener mayor poder de negociación.
  - **Agregación de oferta:** Que permite a las empresas de la CVN poder atender solicitudes de compra que no podrían abastecer individualmente, logrando así mejores oportunidades de negocio.
  - **Información de Interés:** Información de eventos, noticias, grupos de compra/venta, ofertas, demandas a través del portal de la comunidad o mediante dispositivos móviles.

- Servicios ofrecidos por el ambiente de Publicidad y Colaboración (P&C).
  - **Publicación de clasificados**: Permite adicionar y visualizar los eventos a realizarse en la ciudad, ofertas y demandas de productos y servicios, oferta y demandas de empleos; por parte de los usuarios y las empresas.
  - **Publicidad direccionada:** Permite que las empresas puedan realizar publicidad de los productos o servicios que son de interés de un usuario específico.
  - **Salas de discusión con moderador en línea para la realización de reuniones:** Permite la realización de reuniones, en las cuales se establecen unos objetivos y se llega a unas conclusiones, del tema específico que estén tratando.

2.4 Estrategias de Usabilidad y Sociabilidad para la CVN.

Desde la perspectiva del e-commerce el aspecto fundamental del éxito en las comunidades reside en la motivación de las personas para visitar y permanecer en un sitio web comprando bienes o servicios. La manera de hacer negocios da forma a las comunidades dictando su paradigma de interacción social, debido a que Internet y en especial la web es un espacio regido por los negocios[1]. Sin embargo desarrollar comunidades virtuales no es un proceso trivial. Las comunidades virtuales que pretendan ser exitosas, deberán satisfacer las necesidades de sus miembros, al mismo tiempo que contribuyen con el bienestar de la sociedad [6]. Muchas comunidades virtuales nacen y mueren sin dejar rastro. Algunas proporcionan a sus miembros las condiciones para favorecer un valor agregado a sus vidas mientras que otras se desvanecen en el desuso y el olvido. Las comunidades son un proceso que continuamente se desarrolla y evoluciona. Dos estrategias deben considerarse para llevar a cabo el proceso de la comunidad.

2.4.1 Modelo de Usabilidad

Los desarrolladores de software en colaboración con los diseñadores gráficos deben concebir una estrategia de Usabilidad que permita a los miembros de la comunidad interactuar y desarrollar sus tareas de manera fácil e intuitiva. El rápido aprendizaje, la adquisición y retenimiento de las habilidades en el manejo, el bajo número de errores, la alta productividad, la consistencia, predictibilidad y la confortabilidad en el manejo son síntomas de una buena Usabilidad. Este trabajo interdisciplinario se ve reflejado las interfaces gráficas de los ambientes virtuales de P&C, B2C y B2B. A manera de ejemplo se puede observar la Figura 2, en la que se muestra la interface del servicio de Ofertas de la CVN.[6]

Figura 2: Ejemplo de IGU para el servicio de Ofertas de la CVN

2.4.2 Estrategia de Sociabilidad

Aquí se hace énfasis en la interacción social basada en el entendimiento a fondo de las necesidades reales de la comunidad. Esta constituido principalmente por las estrategias, protocolos, políticas, propósitos, procedimientos, planes, indicadores y proyectos que garantizan mediante su operacionalización la puesta en marcha de una identidad común. En la Figura 3, se ilustra la interdependencia de los aspectos de usabilidad y sociabilidad que deben tenerse en cuenta para el desarrollo de una comunidad virtual.[6]

Figura 3: Usabilidad & Sociabilidad

2.6 Puesta en marcha de la CVN bajo la marca Netgociemos.com

Existen desafíos de diversa naturaleza involucrados en la realización y puesta en marcha de la comunidad. Mientras que algunos se corresponden con aspectos tecnológicos o ingenieriles, otros se relacionan más con las personas y su cultura. Si bien los servicios de valor agregado e imagen corporativa de la CVN estuvieron listos para la fecha de culminación del proyecto, algo que definitivamente no se pudo concretar fue la puesta en marcha de la fase piloto de la comunidad, principalmente por dos aspectos fundamentales:

- La imagen corporativa de la comunidad bajo la marca ***Netgociemos.com*** no estuvo completamente integrada al momento de culminar la implementación de los servicios de la plataforma de CVN.
- Se había recopilado información de contacto sobre algunas empresas pero en general fue extremadamente difícil concretar su participación para el final de la ejecución del proyecto (diciembre 25 de 2004), período en que varias empresas desean finalizar sus períodos contables sobre cualquier otra cosa.

En consecuencia y por los motivos arriba planteados, el equipo de desarrollo para la CVN, decidió no poner en riesgo la imagen de la comunidad ***Netgociemos.com***, realizando una presentación rápida de la fase piloto, en su lugar, CYGNUS Tecnologías ha decidido elaborar un diseño cuidadoso de la fase piloto con el propósito de incursionar efectivamente en el mercado regional, posicionando la marca ***Netgociemos.com*** desde un principio con un alto impacto visual, tecnológico y corporativo que propicie el desarrollo de una imagen confiable. Por último, es importante destacar que durante el desarrollo de los ambientes virtuales de la comunidad se realizaron pruebas de desempeño, carga, funcionalidad, estrés entre otras, que garantizaran un funcionamiento sin contrariedades a los usuarios finales de la CVN bajo la marca de Netgociemos.com.

3. PROBLEMAS ENCONTRADOS, EXPERIENCIAS GANADAS Y LECCIONES APRENDIDAS

En los próximos años, los actores principales del comercio electrónico entre empresas serán los e-marketplaces, que son sitios en Internet donde se encuentran compradores y vendedores, para interactuar entre ellos y con el sitio web, conformando con el tiempo comunidades virtuales. El e-marketplace o Comunidad Virtual, para nuestro caso, es el lugar donde las grandes y pequeñas empresas pueden encontrarse, para realizar comercio electrónico entre ellas. Una de las mayores ventajas de pertenecer a una Comunidad Virtual de Negocios, es la Gestión de la Cadena de Suministro (SCM), ya que esta permite tener una visión más amplia del negocio mejorando la coordinación de los procesos al interior de una misma empresa así como entre empresas. Específicamente la CVN ofrece herramientas tecnológicas para el desarrollo de la SCM, que entre otros busca los siguientes beneficios para las empresas del Cauca:

- Disminución de los precios de adquisición de las compras. Con las subastas/bajastas y la agregación de oferta/demanda se tendrán negociaciones más eficaces y eficientes debido a las posibilidades de intercambio de información y transacciones con los proveedores. Igualmente, acceso a mayor número de proveedores potenciales posibilitando el acceso a mayor número de ofertas de manera rápida, sencilla y automática.

- Reducción de costes de compra debido a la eliminación de intermediarios. Debido a la facilidad de intercambio de información entre las distintas empresas, y con la ayuda de las herramientas de B2B, especialmente los servicios integrales y la agregación de oferta y demanda, puede redefinirse la cadena de distribución.

- Disminución de costos en los procesos. Mediante los ambientes de publicidad-colaboración y las herramientas B2B. La sencilla interactividad y colaboración con los proveedores y la integración de los mismos en la cadena de valor permite redefinir los procesos para conseguir una mayor eficiencia y eficacia, y por tanto, la consecuente disminución de costes. Los procesos que se verán más afectados son los relacionados con el intercambio de información y documentación con proveedores, gestión de stocks e inventarios, adquisición de bienes indirectos, gestión de pedidos, logística, etc.

- Ampliación del número de proveedores potenciales y disminución del tiempo de localización. El comprador tiene acceso rápido y económico a gran cantidad de proveedores potenciales tanto a nivel nacional como internacional, teniendo mucha información adicional sobre los mismos que le posibilita su fácil localización y evaluación.

La CVN aprovecha las ventajas que el Internet ofrece con herramientas de almacenamiento de datos y estrategias de bodega de datos y datamining, para ofrecer posibilidades de análisis de información, envío periódico de boletines electrónicos, para mercadeo a la medida, y campañas publicitarias personalizadas que faciliten la recolección y posterior análisis de la información. Elementos que unidos al SCM permitirá que las empresas trabajen para crear experiencias satisfactorias en todos los puntos de contacto del cliente.

A pesar de las difíciles condiciones de violencia y de comunicación vía terrestre en el Cauca, la CVN propone un concepto alternativo de promover-comercializar productos-servicios en la región, al mismo tiempo que impulsa mediante servicios de valor agregado, la asociatividad-colaboración entre empresas miembro de la comunidad, favoreciendo las condiciones iniciales para consolidar mediante un ambiente virtual, un punto de encuentro entre empresas y clientes, promoviendo canales alternativos de comunicación. Un aspecto fundamental en el desarrollo de la CVN, fue la investigación de mercados que permitió dimensionar la expectativa de los servicios que debían ofrecerse a la comunidad, al mismo tiempo que perfiló la estrategia de liberación de los mismos. Además la CVN establece los mecanismos de sostenibilidad y su posterior regulación en el marco de la legislación colombiana, durante su puesta en marcha. La alta disponibilidad de parte de los empresarios de la región, y una adecuada sensibilización que muestre las ventajas competitivas del uso de las TI, garantizará la puesta en marcha de la CVN. El proyecto CVN incorpora un alto impacto en la economía de la región caucana, por lo tanto, es importante que iniciativas como ésta se sigan apoyando, permitiendo que las oportunidades que ofrecen la innovación tecnológica y las TI se ponga al servicio de las empresas de la región, mejorando su competitividad e incrementando su sosteniblilidad y participación en el mercado.

La CVN constituye el proceso que nace y evoluciona como proyecto, sin embargo para hacer realidad el impacto al sector productivo del Departamento del Cauca, debe transferirse CVN como producto a través de la creación de una nueva empresa que administre, gestione, comunique, socialice y evolucione CVN en su etapa de producción. Esta organización ha nacido con el nombre de ***Netgociemos***[4]. La puesta en marcha de esta nueva empresa caucana, requiere seguir el plan estratégico, plan de negocios, modelo de sociabilidad y marco legal definidos para ella[4], para de esta forma operar adecuadamente las exigencias sociales y tecnológicas derivadas de la administración de la nueva comunidad virtual.

Planificar cuidadosamente la interacción social, la funcionalidad y usabilidad del software no garantiza el éxito de una comunidad en línea, sin embargo, la comunidad fracasaría si no se tienen en cuenta estos principios críticos, los cuales, establecen la base para orientar y enfocar a los desarrolladores de comunidades virtuales en el alcance de objetivos de la comunidad virtual. Las comunidades son dinámicas, evolucionando y cambiando constantemente, todo el tiempo sufren las influencias causadas por las diversas personalidades de sus miembros, las mismas actividades de la comunidad y algunas veces hasta reciben influencias externas. De otro lado, los aspectos tecnológicos no deben convertirse en la preocupación fundamental de una comunidad en línea, en su lugar, las personas o miembros de la comunidad deben considerarse como la preocupación central, hacia la cual deben enfocarse los esfuerzos de mejora y cambio. Por último, los desarrolladores, tienen muy poco o nada de control sobre los miembros de la comunidad, sin embargo, ellos pueden garantizar un buen comienzo para la comunidad, en primer lugar, mediante el desarrollo de software confiable, funcional y de fácil uso y en segundo lugar, mediante la planificación cuidadosa de las estructuras sociales de la comunidad, en otras palabras el modelo o estrategia de sociabilidad de la comunidad.

Agradecimientos
Los autores de este artículo, desean expresar sus agradecimientos a las siguientes entidades y personas, que realizaron sus aportes en recursos, talento humano, tecnología e infraestructura, y sin los cuales hubiera sido imposible llevar a cabo este trabajo:

1. COLCIENCIAS. Entidad Gubernamental Co-Financiadora del Proyecto.
2. UNIVERSIDAD DEL CAUCA. Entidad de Educación Superior Co-Financiadora del Proyecto.
3. CYGNUS TECNOLOGÍAS. Empresa privada Co-Financiadora del Proyecto.
4. HEXUM Medios Digitales.

Referencias

[1] Jones, Q., Rafaeli, S. User population and user contribution to virtual publics: A systems model. *Paper presented at the supporting Group Work (Group´99)*, Phoenix, AZ., 1999.

[2] Larman, C. UML y Patrones. 2da. Edición, Pearson Education. ISBN: 8420534382.

[3] Mendoza, M., Sierra, L., Naranjo, R., Moreno, J., Cygnus Tecnologías. Propuesta "Comunidad Virtual de Negocios –CVN para el Departamento del Cauca, Plataforma Comercial en Internet." Mayo 2 de 2002.

[4] Mendoza, M., Sierra L., Naranjo, R., Moreno, J. Cygnus Tecnologías. Informe Técnico Final del proyecto -"Comunidad Virtual de Negocios –CVN para el Departamento del Cauca, Plataforma Comercial en Internet." Presentado a COLCIENCIAS. Diciembre de 2004.

[5] Moreno, J., Naranjo, R., Sierra, L., Mendoza M. Building A Virtual E-Commerce Community. IADIS International Conference e-Commerce 2004. Portugal. December 2004.

[6] Preece, J. Online Communities "Designing Usability, Supporting Sociability. *Jhon Wiley and Sons Ltda*. Marzo de 2001.

Power and Real-Time Requirements Integrated with an Adaptive Resource Reservation Soft Real-Time Scheduler

Rodrigo Santos, M. Belén D´Amico, Javier Orozco
Universidad Nacional del Sur-CONICET,
Depto. de Ingeniería Eléctrica y de Computadoras,
Bahía Blanca, Argentina, B8000CPB
{ierms,mbdamico}@criba.edu.ar

and

Pedro D. Doñate
Universidad Nacional del Sur,
Depto. de Ingeniería Eléctrica y de Computadoras,
Bahía Blanca, Argentina, B8000CPB
pdonate@criba.edu.ar

and

Leo Ordínez y **David Donari**
Universidad Nacional del Sur,
Depto. de Ciencias e Ingeniería de la Computación,
Bahía Blanca, Argentina, B8000CPB

Abstract

Recently, devices such as cellular phones, portable computers and digital agendas have experienced a great development. The commercial success of these products greatly depends on the battery life. Although batteries have improved considerably, microprocessors are more and more power-hungry. This paper presents a new alternative for controlling energy consumption in soft real-time applications. A resource reservation paradigm working under Earliest Deadline First scheduling is applied. The bandwidth reserved is determined by using an adaptive predictor and a feedback algorithm. This dynamical adjustment permits to scale the frequency and voltage of the processor, reducing the consumed energy. The usefulness of the proposed technique is illustrated via experimental results.

Keywords: Open Systems, Real-Time, Power-Aware

Resumen

En los últimos años, dispositivos electrónicos como los telefónos celulares, las agendas electrónicas y las PC portátiles han evolucionado considerablemente. El éxito de esta clase de productos depende fundamentalmente de la duración de sus baterías. A pesar que el rendimientos de las mismas mejora continuamente, la demanda de energía de los procesadores cada vez mayor. En este trabajo se presenta una nueva alternativa para controlar el consumo de energía en aplicaciones de tiempo real. El paradigma de reserva de recursos planificado con Menor Tiempo al Vencimiento es utilizado. El ancho de banda reservado se ajusta empleando un predictor y un filtro Proporcional Integral en el lazo de realimentación. Este procedimiento permite variar la frecuencia y el voltage de alimentación del procesador disminuyendo así la energía consumida. Resultados experimentales ilustran la utilidad de la técnica propuesta.

Palabras claves: Sistemas Abiertos, Tiempo Real, Ahorro de Energía

1 INTRODUCTION

The continuous evolution of electronics and the increasing possibility of integration have permitted the creation of a completely new family of applications working on real time. Fortunately, most of them are not critical and can tolerate the occasional miss of some deadlines. Some common examples are multimedia streaming programs, video/audio players, software sound mixers, cellular phones, etc. Since these applications are very often implemented on portable devices, they should be designed to maximize battery life. However, they usually need power-hungry processors that consume a great amount of energy.

Recently, important advances in the study of the energy problems in portable devices have been reported. The *Dynamic Voltage Scaling* (DVS) is a technique that benefits from two facts:i) processors are not always working at the maximum of their capacity; and ii) they are built on CMOS logic. The idea is to reduce the consumed energy scaling both frequency and voltage of operation of the processor whenever there is no need for high performance. However, the basic trade-off between performance and consumption of energy is still an open issue.

The quality of service (QoS) of mobile real-time applications can be measured by the amount of missed deadlines. Usually, real-time systems are composed of a set of tasks sharing many resources like CPU, memory, disks, device drivers, I/O devices, etc. The operating system has to assign the processor and the rest of the resources to the tasks in such a way that they can meet their deadlines. In [24], a scheduling paradigm called *resource reservation* is presented. Under this approach, each task makes a bandwidth reservation of the CPU. This means that they are executed only for a certain amount of time (budget) periodically. Once the budget is depleted or any of the tasks finished, the processor is assigned to another one. In this way, tasks execute on virtual processors with a speed proportional to the bandwidth allocated to the servers. Overruns do not affect other tasks since they are temporally isolated.

The resource reservation mechanism strongly depends on the correct allocation of CPU bandwidth to each task. If tasks present wide variations in the required computation time, a static choice for the bandwidth might produce deficient results: it could exist a plenty of idle time or, on the contrary, it could be necessary more time to avoid a deadline miss. Choosing the worst case execution time leads to a pessimistic design forcing an overestimated hardware for the majority of the cases. To solve this problem, feedback control mechanisms is introduced in [2]. The authors show that a CPU reservation can realistically be modeled as a switching and parametric dynamical system (the computation time for each activation of the task acts as the varying parameter). Therefore, a controller can be designed and the bandwidth for each server can be adjusted dynamically to the task´s demand. The limitation of this kind of solutions based on traditional linear controllers or *ad hoc* nonlinear ones is that they cannot anticipate the variations on the execution times of the tasks. Recently, a simple moving average predictor plus an stochastic dead beat feedback controller have been proposed to improve the overall performance of the system. However, this predictor does not consider that many of this signals have a certain degree of correlation [7].

In this paper, a new method to control the consumption of energy in processors is presented. Since the resource reservation paradigm is applied under the general scheduling policy of Earliest Deadline First, based on DVS, the processor operating frequency is varied in such a way that the bandwidth demanded is always close to 1. In this sense, the system operates like a full loaded, low power and reduced performance processor fulfilling the QoS. Each task in the system is allocated to a dedicated server. The budget of the servers (bandwidth) is controlled via an adaptive predictor based on the Least Mean Square (LMS) algorithm. It anticipates the changes in the execution times. A Proportional Integral controller is also used in the feedback loop to compensate for wrong assignments. Both the predictor and the feedback controller have sense only in the scope of physical process where execution times conform a weakly stationary stochastic process. Tasks are assumed to be independent preventing in this way blockings in shared resources.

In Section 2, related work is revised and compared to the new approach proposed in this paper. In Section 3, the dynamical model of the tasks, the resource reservation and the frequency/voltage variations are presented. In Section 4, the predictor and controller design are described. In Section 5, the usefulness of the technique is illustrated via experimental evaluations. Finally, conclusions are drawn in Section 6.

2 PREVIOUS WORK

The continuous miniaturization and development of portable devices originate an important problem in the duration of battery life. To solve this kind of inconvenience the well-known DVS technique is usually employed

[3, 9, 11, 15, 18, 22]. This method is based on the fact that the peak demand of most computing systems (designed nowadays on CMOS logic) is much higher than the average one. The previous characteristics permits the reduction of the frequency according to the actual demand of the system. Furthermore, as the common logic is CMOS, it is also possible to reduce the voltage of operation, leading to considerable savings in the energy consumed by the processor.

For devices supporting real-time applications, the DVS should be carefully applied since the decrease of the frequency of the processor enlarges the execution time of the task. This problem can be overlooked in certain applications like portable digital agendas but not in cellular phones or other specific devices. In [20], a new approach to DVS, named RT-DVS, is introduced. The authors present two versions based on the utilization factor of the system and static/dynamic voltage-frequency reductions. The dynamical part is considered whenever a task executes in less time than its worst case, generating idle time. Nevertheless, the method does not include a control algorithm to provide a tracking of the bandwidth demand on the processor. There are many other papers that use the DVS method to save energy on CMOS processor based systems. As an example, an algorithm to determine the optimal frequency and voltage of operation on fixed priority driven systems is presented in [23]. However, this approach does not consider the possibility of working on line on an open dynamic environment with tasks entering and leaving the system.

Basically, there exist two different approaches to deal with tasks that have great variations on the execution times in successive instances. The first one, developed in [10, 27, 25], studies the schedulability of the system from an stochastic point of view. Instead of working with the worst case execution times (making the analysis too pessimistic), they work with probability density functions of the execution times. Hence, they obtained an statistical characterization of the time response of the task and the deadline miss probability as figure of merit for the system. This technique however is unable to predict how a missed deadline could affect other tasks and besides it has no control over the bandwidth requirement. The second approach includes the utilization of resource reservation mechanisms via servers that provide temporal isolation between different tasks. Each task assigned seems to run on a virtual processor with a speed proportional to the bandwidth allocated to its respective server. This introduces a hierarchical scheduling policy: in the first level, the Operating System dispatches the servers while, in the second level, the different servers schedule the execution of tasks allocated to them. There are many kind of servers described in the literature: Total Bandwidth Server [4], Constant Utilization Server [8], Constant Bandwidth Server (CBS) [1], etc. In the particular case of the CBS, a statistical guarantee for soft real time tasks and a deterministic one for hard tasks are presented. All of these treatments are very useful in open dynamical systems where the set of tasks is not defined previously and there is poor knowledge about their main parameters (the worst case execution time and the minimum inter arrival time between successive instances of the same task).

Many efforts have been made to model real time schedulers as linear dynamical systems in order to apply the traditional controllers and optimize their performance. Some of the ideas are focused on the sampling frequency of variables or task´s periods to cope with overruns on other tasks [6, 5]. In [17], an analytical framework to map QoS requirements of adaptive real time systems into feedback control theory is proposed. However, this feedback control scheduler does not use a resource reservation paradigm. In [2], a feedback controller to adapt the budget or bandwidth of a CBS based on a Proportional plus Integral (PI) controller is introduced. The choice of a PI controller is adequate when signals vary in a step wise form. This work is extended in [19] where a nonlinear approach based on some statistical characterization of the execution time of the task is considered. The results obtained in this case are more accurate than those presented previously. Finally, an integral approach consisting of a predictor based on a Moving Average Filter and an Stochastic Dead Beat controller for the feedback loop is developed in [7].

The aim of this paper is to fulfill the logical and electrical requirements of the system regarding two concepts: schedulability and energy saving. Tasks are allocated to servers and bandwidths are controlled via a LMS adaptive predictor and a proportional integral feedback mechanism. The adjusted bandwidth allow us to reduced both frequency and voltage of operation, achieving a significant reduction in the energy consumed by the system.

3 DYNAMICAL MODEL OF THE SYSTEM

Tasks are considered to be periodic and independent. A task τ_i can be seen as a stream of jobs or instances J_{ij} where i and j stand for task and instance, respectively. Jobs are released periodically; the instant at which they become ready for execution, notated r_{ij}, is $r_{ij} = r_{ij-1} + T_i$, and the corresponding deadline is

computed as $d_{ij} = r_{ij} + T_i$, where T_i is the period of the task. The finishing time of an instance is notated f_{ij}. If $f_{ij} \leq d_{ij}$ the job finishes within its deadline. Each job has an execution time c_{ij} that varies in the interval $[C_{min}, C_{max}]$. In general, c_{ij} is related to some physical variable and is not an independent, identically distributed random variable. On the contrary, it has a certain degree of correlation.

In the following, time is considered to be slotted and the duration of one slot is taken as the indivisible unit of time. Slots are notated as t and numbered 1, 2, 3,... .The expressions at the *beginning of slot t* and *instant t* are equivalent. Lower priority tasks can be preempted by higher priority ones only at the beginning of the slots. An *empty* slot means that there is not a task ready to be executed during that unit of time.In that case the processor becomes idle [26].

3.1 Resource reservation mechanism

The resource reservation mechanism is implemented by using the CBS mechanism [1]. Each task τ_i in the system is allocated to a determined CBS notated as S_i. The server S_i is described by a pair (Q_i, P_i) where Q_i is the budget and P_i is the period of the server. Basically, the server guarantees Q_i units of time for running every allocated period P_i. The relation Q_i/P_i indicates the utilization factor or bandwidth (B_i) of the server. Whenever two servers have active tasks to dispatch, the one with the highest priority is chosen. Priorities can be assigned to servers based on some scheduling policy like Rate Monotonic (RM) or Earliest Deadline First (EDF). In this paper, the system is scheduled by EDF; thus the maximum utilization factor achievable by the processor is 1 [16]. The necessary and sufficient condition for the schedulability of the system under EDF is that the sum of the utilization factors should be less than 1, $\sum_i Q_i/P_i \leq 1$.

Each server has two dynamical variables, q_i and δ_i. The first one is the current budget and keeps track of the consumed budget. The second one is the scheduling deadline. Initially, q_i is equal to Q and then it is decremented by one each time the server executes one slot. When it reaches zero, the budget is recharged and the deadline is postponed. The rules for governing the CBS are the following:

Rule A: When a job J_{ij} of task τ_i arrives at time r_{ij}, the server checks the following condition $q_i \leq Q_i(\delta_i - r_{i,j})/P_i$. If the inequality is verified, the current pair (q_i, δ_i) is used. Otherwise, a new pair (q_i, δ_i) is computed, *i.e.* $q_i := Q_i$ and $\delta_i := r_{ij} + P_i$.

Rule B: If server S_i executes for Δt slots, the budget is decremented accordingly $q_i := q_i - \Delta t$.

Rule C: Server S_i is allowed to execute while $q_i \geq 0$. When the budget is depleted, a new pair (q_i, δ_i) is computed: the scheduling deadline is postponed to $\delta_i := \delta_i + P_i$ and the budget is recharged to $q_i := Q_i$. Since the scheduling deadline has changed, the EDF queue has to be reordered, and preemption may occur.

The dynamical behavior of a resource reservation system is modeled in [2]. For completeness, some of the definitions and basic equations are reproduced here.

The *virtual finishing time* of an instance is the time necessary to finish its execution if it runs on a dedicated processor with speed B_s, *i.e.*

$$VFT_{ij} = \frac{c_{ij-i}}{B_i}. \tag{1}$$

The *Latest Possible Finishing Time* is the number of reservation periods used by the job to complete its execution, *i.e.*

$$LFT_{ij} = \left\lceil \frac{c_{ij-1}}{B_i P_i} \right\rceil P_i. \tag{2}$$

There is a clear relation between VFT and LFT,

$$VFT_{ij} = LFT_{ij} - \frac{q_i}{B_i}. \tag{3}$$

The last term in (3) accounts for the remaining budget in the server. As can be appreciated, $LFT_{ij} \geq VFT_{ij}$.

The *scheduling error* is defined as $e_{ij} = VFT_{ij} - T_i$.

The substitution of VFT_{ij} by LFT_{ij} is a conservative but secure option to measure the scheduling error. If $LFT_{ij} \geq T_i$, the bandwidth assigned to the CBS holding the task is not enough and thus, it has to be incremented. Moreover, the extra time used will interfere with the next job of the task.

Based on these definitions the dynamics of the reservation system for each server can be modeled as

$$e_k = \begin{cases} e_{k-1} + \frac{c_{k-1}}{B_{k-1}} - T & e_{k-1} \geq P, \\ \frac{c_{k-1}}{B_{k-1}} - T & e_{k-1} < P. \end{cases} \tag{4}$$

Some subindexes have been removed for clarity. The interested reader can find a complete description of the equations and their deductions in [2].

To linearize the system, variable B_k is changed by its reciprocal, $u_k = 1/B_k$. Therefore, the above equation can be expressed as

$$e_k = \begin{cases} e_{k-1} + c_{k-1}u_{k-1} - T & e_{k-1} \geq P, \\ c_{k-1}u_{k-1} - T & e_{k-1} < P. \end{cases} \tag{5}$$

Considering that the system evolves in a very small neighborhood around its equilibrium point, (5) can be transformed into

$$\triangle e_k = \begin{cases} \triangle e_{k-1} + \overline{c} \triangle u_{k-1} + \overline{u} \triangle c_{k-1} & e_{k-1} \geq P, \\ \overline{c} \triangle u_{k-1} + \overline{u} \triangle c_{k-1} & e_{k-1} < P. \end{cases} \tag{6}$$

The $\triangle$ operator can be omitted to keep the notation simple. Hence, the final representation of the system is

$$e_k = \begin{cases} e_{k-1} + \overline{c}u_{k-1} + \overline{u}c_{k-1} & e_{k-1} \geq P, \\ \overline{c}u_{k-1} + \overline{u}c_{k-1} & e_{k-1} < P. \end{cases} \tag{7}$$

3.2 Voltage-frequency scaling

In CMOS processors power can be expressed as $P = kV_{DD}^2 C_L f$, where V_{DD} is the voltage from the power source, C_L is the capacitance in the circuit, f is the operation frequency and k is a proportional constant [20]. In this paper, a DVS is applied to enhance the battery life of the system. Based on the bandwidth demand of the servers the frequency is reduced in order to have an utilization factor close to 1. Since the scheduling policy chosen is EDF, deadlines are guaranteed whenever the utilization factor is less than 1. Frequency is varied in accordance to the following

$$\sum_{i=1}^{n} \frac{Q_i}{P_i} = \alpha < 1 \Rightarrow f_{new} = f_{nom} \cdot \alpha \tag{8}$$

where f_{nom} is the nominal frequency of the processor and $0 < \alpha \leq 1$.

As was mentioned before, the frequency of the CMOS circuits is proportional to the voltage. Therefore, a reduction in the frequency implies a reduction in the voltage.

$$\begin{aligned} f_{nom} &= \beta V_{DD(nom)},\ f_{new} = \alpha f_{nom} \\ &\Rightarrow f_{new} = \alpha\beta V_{DD(nom)} \end{aligned} \tag{9}$$

where β is a proportional constant that relates the actual frequency with the voltage. From (8) and (9) it is possible to compute the power reduction due to the reduced frequency and voltage.

$$P_{nom} = kV_{DD(nom)}^2 C_L f_{nom} = k\frac{C_L}{\beta^2} f_{nom}^3$$

$$P_{new} = k$$

$$E_{nom} = P_{nom} \cdot t_{nom} = k\frac{C_L}{\beta^2} f_{nom}^3 t_{nom}$$

Whenever the frequency of operation of the processor is reduced, the time required to finish the same task is increased proportionally, that is $t_{new} = \alpha^{-1} t_{nom}$. Thus,

$$E_{new} = k\frac{C_L \alpha^3}{\beta^2} f_{nom}^3 \frac{1}{\alpha} t_{nom}$$

$$\frac{E_{new}}{E_{nom}} = \alpha^2 \tag{10}$$

The previous reasoning shows that a reduction in frequency produces a quadratic reduction in the energy consumed by the processor.

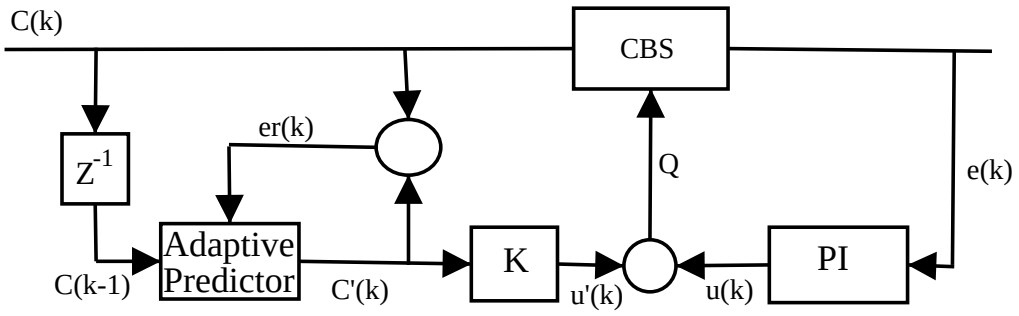


Figure 1: Block Diagram of the System

4 CONTROLLER DESIGN

The proposed control strategy can be divided in two parts. One of them consists of an adaptive predictor that estimates the value of the execution time of the next instance based on the execution times of previous instances. If there is a considerable difference in the estimation and the bandwidth allocated by the predictor is not enough, a scheduling error will be generated and a deadline could probably be missed. To compensate as quickly as possible this deficit, a proportional integral feedback controller is introduced. Figure 1 shows the general block diagram of the system for one task/CBS.

4.1 Adaptive predictor based in the LMS algorithm

Assuming that the execution time c_k is a weakly stationary correlated stochastic sequence over short periods of time (*i.e.*, the sequence exhibits dependence between different time samples and also some stationary statistical properties over certain intervals associated with different tasks), its value can be estimated ahead in time by using a simple finite impulse response (FIR) *adaptive predictor*. As will be shown, the adaptation characteristic of the filter is also necessary to track changes in the mean and correlation of the sequence.

The output of the predictor is given by

$$\widehat{c}_k = \sum_{i=0}^{N} \theta_{i(k)} c_{k-i-1} = \mathbf{\Theta}_k^T \mathbf{C}_k \tag{11}$$

where N is the number of coefficients θ_i of the FIR,

$$\mathbf{C}_k = \left[c_{0(k-1)}, c_{1(k-2)}, \ldots, c_{N(k-N-1)}\right]^T$$

is the regressor and $\mathbf{\Theta}_k = \left[\theta_{0(k)}, \theta_{1(k)}, \ldots, \theta_{N(k)}\right]^T$ is the coefficient vector. From (11), the prediction error is

$$er_k = c_k - \widehat{c}_k = c_k - \mathbf{\Theta}_k^T \mathbf{C}_k. \tag{12}$$

Then, the coefficients of the FIR are obtained and updated by an algorithm that minimizes a function of the prediction error er_k. That function can be, for instance, the mean square error (MSE)

$$J(\Theta) = E[er_k^2], \tag{13}$$

where $E[\cdot]$ is the expectation operator. Replacing (11) and (12) into (13), results

$$J(\mathbf{\Theta}) = E\left[c_k^2\right] - 2\mathbf{\Theta}_k^T \mathbf{U}_k + \mathbf{\Theta}_k^T \mathbf{R}_k \mathbf{\Theta}_k, \tag{14}$$

where $\mathbf{U}_k = E[c_k \mathbf{C}_k]$ and $\mathbf{R}_k = E\left[\mathbf{C}_k \mathbf{C}_k^T\right]$. A gradient based family of adaptive algorithms can be generated by using a coefficient updating equation of the form

$$\mathbf{\Theta}_{k+1} = \mathbf{\Theta}_k - \mu \mathbf{G}_k, \tag{15}$$

where

$$\mathbf{G}_k = \frac{\partial J(\theta)}{\partial \theta} = 2\left(\mathbf{R}_k \mathbf{\Theta}_k - \mathbf{U}_k\right) \tag{16}$$

is the gradient vector of (14) in the coefficients space and μ is a small positive number that acts as a convergence factor. Different approaches for the evaluation of an estimate of the real theoretical gradient $\mathbf{G}_k$ result in different algorithms. The Least Mean Square (LMS) algorithm [28, 12], uses the instantaneous values of $\mathbf{U}_k$ and $\mathbf{R}_k$ as estimates of their means, *i.e.*,

$$\widehat{\mathbf{U}}_k = c_k \mathbf{C}_k \tag{17}$$

and

$$\widehat{\mathbf{R}}_k = \mathbf{C}_k \mathbf{C}_k^T. \tag{18}$$

Substituting (17) and (18) in (16), Equation (15) for updating the coefficients transforms into

$$\mathbf{\Theta}_{k+1} = \mathbf{\Theta}_k - 2\mu e r_k \mathbf{C}_k.$$

The complexity of this predictor is $O(2N^2)$. Since the correlation is usually limited to a few number of samples (N is less than 5), it is possible to implement it on line.

4.2 Proportional Integral Controller

Considering that the disturbances in the execution times have an step wise form, a proportional integral controller can be used to cancel the consequent stationary error. The equation describing the PI controller is

$$u_k = u_{k-1} - \beta e_k - \gamma e_{k-1}.$$

A complete description for the design of a PI controller applied to system (7) and the corresponding stability analysis can be found in [2]. The interested readers can also consult [21], for more general information about digital PI controllers. The PI parameters β and γ could be determined by choosing the appropiate location of the poles for the closed-loop transfer function of the system. However, if faster responses are desired to keep the average bandwidth as lower as possible, higher transient bandwidth values will be necessary to compensate the possible disturbances. For the purpose of this paper, the poles have been located at 0.1 and 0.7. These values represent a good trade off between transient response and bandwidth requierement.The output of the controller is added to the predicted value and the result is introduced in the CBS, as it is shown in Figure 1.

4.3 Restrictions

1. Although the feedback controller and the predictor can produce any real value, the budget of the server is constrained to the natural numbers. This introduces a restriction in the output of the controller. To be on the safe side, the ceiling operator has to be used.

2. The controller should always produce a budget smaller than the period of the respective server. If not, the utilization factor of the system will be greater than one.

3. If there exist more than one server in the system, the upper bound on the budget is given by $\sum_{i=1}^{n} Q_i/P_i \leq 1$. Since the updates of the budgets are not synchronous, and it is not possible to modify the reservation already done by another server, the scaling has to be done over the current server. Suppose that server S_j is being updated and that the sum of the utilization factors exceeds 1, then

$$\sum_{i=1}^{n} \frac{Q_i}{P_i} = W > 1 \Rightarrow Q_j = P_j \left(1 - \sum_{i=1, i\neq j}^{n} \frac{Q_i}{P_i}\right). \tag{19}$$

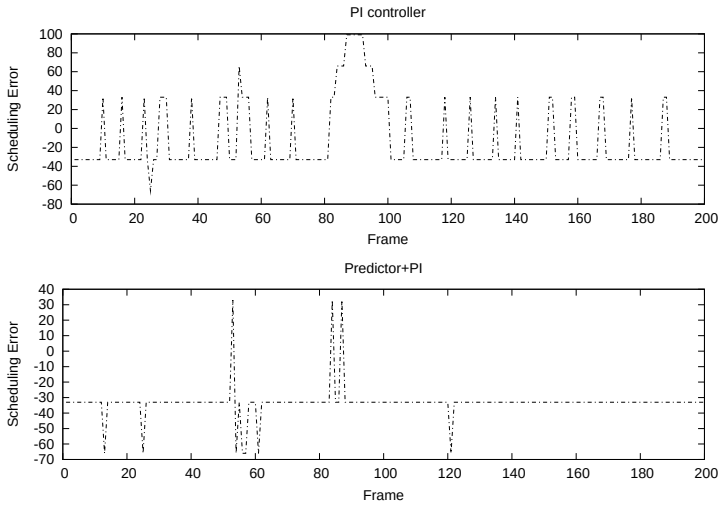


Figure 2: Scheduling Error

4.4 Voltage-Frequency Control

In practice, a processor has a finite number of pairs (f, V) to choose from. Thus, based on (8) the closest pair has to be chosen. The experimental evaluation presented in the next section does not contemplate the granularity of frequency and voltage since it is dependent on the kind of processor the system will be implemented.

5 EXPERIMENTAL EVALUATION

The performance of the proposed algorithm was evaluated via several experimental simulations. Some mpeg movies were downloaded from the web. The code of the smpeg player for LINUX was modified in such a way that the decodification and display time for each frame could be computed. In order to eliminate the possible interference of other threads running under the Linux OS, the movie was played back ten times and the execution times averaged.

The examination was divided into two parts. In the first one, the dynamical behavior of the system was simulated using Octave [29]. The difference equations corresponding to the CBS, the predictor and the PI controller of only one task were implemented. Resembling the configuration presented in [2], the same system but without the predictor was also simulated. The file containing the execution times of the movies were used to feed the programs. For each sample in the file, the scheduling error, the bandwidth as well as the optimal execution frequency of the processor were computed. The predictor was designed to work with a history of three samples. As an example, the results obtained with a real movie are shown in Figures 2 and 3.

In Figure 2 the scheduling error for 200 frames is presented for both algorithms. Every time the scheduling error is greater than zero, a deadline is missed. As can be seen the utilization of the predictor and the feedback controller reduces the number of deadline misses providing a better QoS for the playback of the movie. Figure 3 shows that the bandwidth obtained by using the PI plus the predictor presents in general small variations. This behavior change drastically when the predictor is excluded. In this case, the curve has a sawtooth form. Moreover, the mean bandwidth increase from 0.38 to 0.56.

In the second set of experiments, the temporal evolution of the system with four tasks was simulated using a program written in C++. The scheduler, the tasks, the controller and the predictor for each CBS was implemented. The worst case utilization factor for the system is well above 1. However, the use of the RR paradigm and the adaptive bandwidth allows a good use of the processor even for an overload situation like this. The tasks arrive to the system at different times so the controller and predictor for each one are not synchronized. Figure 4 shows the bandwidth consumed by the processor when the four tasks are running.

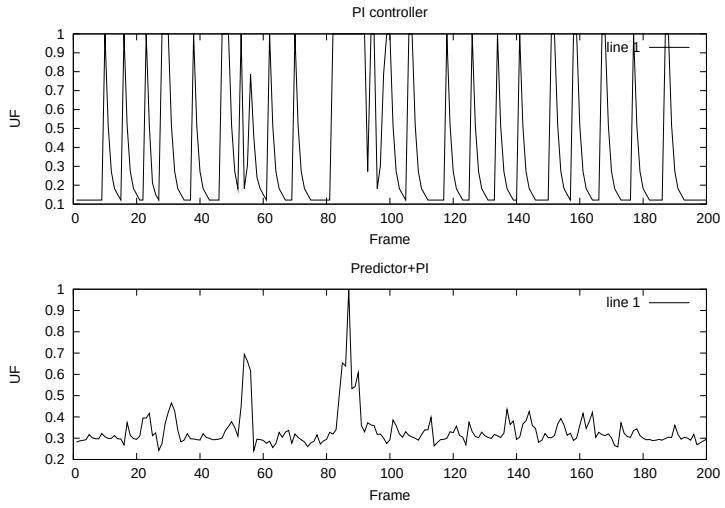


Figure 3: Bandwidth assigned

The first task is used for sampling the processor state.

Taking into account the average bandwidth obtained with both methods, it is possible to compute the energy consumption of the system, given by (10). Hence, the ratio of saved energy in both cases is:

$$\frac{E_{P+PI}}{E_{nom}} = 0.144,\ \frac{E_{PI}}{E_{nom}} = 0.314$$

The introduction of the predictor for the bandwidth reservation produces an extra saving of near 50% with respect to the simple PI feedback. The quality of service measured as the number of deadlines missed is also incremented. In other words, the experimental evaluation shows an overall better performance of the algorithm proposed here.

6 CONCLUSIONS

A new approach to reduce the energy consumption of CMOS processors was presented. The mechanism is based on the CBS paradigm to temporally isolate the behavior of the different tasks running on the system. In this way, it is possible to include an adaptive predictor and a PI controller to assign the necessary bandwidth to each task dynamically. This adjustment also permits to scale the frequency and voltage of the processor, reducing the consumed energy. The proposed technique was implemented using Octave and C++ programs. The results show that the combination of a simple predictor plus a PI filter can improve the general performance of portable devices. However, its usefulness is restricted to soft real-time applications.

In the case the processor has to manage a hard real-time application (where the lost of a deadline is not allowed), it is possible to provide a deterministic guarantee of success assigning to the respective server the maximum bandwidth corresponding to the worst case execution time. The rest of the tasks can be controlled using the adaptive bandwidth approach.

In the near future, the authors will work on a real-time operating systems to implement the method on a real platform. Some of the problems to solve are the granularity in the frequency and voltage variation. The options to implement the module are RT-Linux [14] or S.Ha.R.K. [13].

Acknowledgments

The authors appreciate the financial support of SGCyT at the Universidad Nacional del Sur and CONICET. Maria Belén D'Amico also acknowledges the support of ANPCyT (PICT -11- 12524).

References

[1] Abeni L., y G. Buttazzo, "Integrating multimedia applications in hard real-time systems", *Proc. 19th IEEE Real Time Systems Symposium*, Madrid, December 1998.

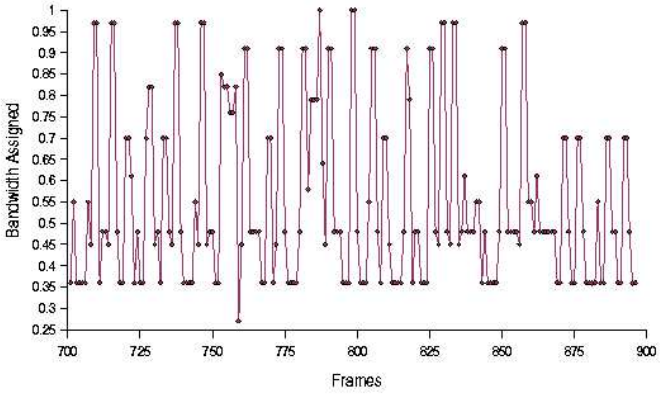


Figure 4: Bandwidth Assigned for 4 tasks

[2] Abeni L., L. Palopoli, G. Lipari, J. Walpole, "Analysis of a reservation feedback scheduler", *Proc. 23rd IEEE Real Time Systems Symposium*, 2002.

[3] Burd, T. D. and Brodersen, R. W., "Energy efficient CMOS microprocessor design", *Proc. 28th Annual Hawaii Intl. Conf. on Systems Sciences, Vol.1: Architecture, IEEE Computer Soc. Press*, January 1995.

[4] Butazzo, G. and Sensini, F. "Optimal deadline assignment for scheduling soft aperiodic tasks in hard real-time environments", *Proc. 3rd. Intl. Conf. on Engineering of Complex Computer Systems*, pp. 39-48, September 1997.

[5] Buttazzo, G., G. Lipari, M. Caccamo, L. Abeni, "Elastic Scheduling for Flexible Workload Management", *IEEE Transactions on Computers*, Vol. 51, No. 3, pp. 289-302, March 2002.

[6] Caccamo, M., G. Buttazzo, and L. Sha, "Elastic Feedback Control", *IEEE Proc. 12th Euromicro Conference on Real-Time Systems*, Stockholm, Sweden, pp. 121-128, June 2000.

[7] Cucinotta, T., Palapoli L., Marzario L., "Stochastic feedback-based control of QoS in soft real-time systems", *Proc. of Control Decision Conference*, Bahamas, December 2004.

[8] Deng Z., and J. W. S. Liu, "Scheduling real-time applications in open environment", *Proc. 18th IEEE Real Time Systems Symposium*, San Francisco, December 1997.

[9] Flautner, K., Reinhardt, S. and Mudge, T., "Automatic performance-setting for dynamic voltage scaling", *Proc. 7th Conf. on Mobile Computing and Networking MOBICOM*01, *July*2001.

[10] Gardner, M., *Probabilistic analysis and scheduling of critical soft real-time systems*, doctoral diss, University of Illinois at Urbana-Champaign, 1999.

[11] Gruian, F., "Hard real-time scheduling for low energy using stochastic data and DVS processors", *Proc. of the Intl. Symposium on Low-Power Electronics and Design ISLPED*01, *August*2001.

[12] Haykin, S., Adaptive Filter Theory, Prentice Hall Inc., third edition, 1996.

[13] http://shark.sssup.it

[14] http://www.ocera.org

[15] Krishna, C. M. and Lee, Y. H., "Voltage-clock-scaling techniques for low power in hard real-time systems", *Proc. IEEE Real-Time Technology and Applications Symposium*, May 2000.

[16] Liu C. L. and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal ACM*, Vol. 20, No. 1, (1973), pp. 46-61.

[17] Lu, C., J. Stankovic, G. Tao and S. Son, "Feedback Control Real-Time Scheduling: Framework, Modeling and Algorithms", special issue of *RT Systems Journal on Control-Theoretic Approaches to Real-Time Computing*, Vol. 23, No. 1/2, (July/September, 2002), pp. 85-126.

[18] Mosse, D. Aydin, H. Childres, B. and Melhem, R., "Compiler-assited dynamic power-aware scheduling for real-time applications", *Workshop on Compilers and Operating Systems for Low-Power (COLP*00), *October* 2000.

[19] Palopoli, L., Luca Abeni and Giuseppe Lipari, "On the applications of hybrid control to CPU Reservations", *Proc. of Hybrid systems Computation and Control HSCC03* (Lecture notes on computer science), Prague, The Czech Republic, April 2003.

[20] Pillai, P. and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," *Operating Systems Review*, Vol. 35, No. 5 (October 2001), pp. 89-102.

[21] Phillips, C. L. and H. T. Nagle, *Digital Control System: Analysis and Design*, Prentice Hall Inc., New Jersey, 1990.

[22] Pouwelse, J., Langendoen, K. and Sips, H., "Dynamic voltages scaling on a low-power microprocessor", *Proc. 7th Conf. on Mobile and Computing and Networking MOBICOM01*, July 2001.

[23] Quan, G. and X. (Sharon) Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors", *Proc of IEEE/ACM Design Automation Conference*, pp 828-833, June 2001.

[24] Rajkumar, R., K. Juvva, A. Molano and S. Oikawa, "Resource kernels: A resource-centric approach to real-time and multimedia systems", *Proc. SPIE/ACM Conf. on Multimedia Computing and Networking*, January 1998.

[25] Santos R., J. Santos, J. Orozco, "Hard real-time systems with stochastic execution times: deterministic and probabilistic guarantees", *International Journal of Computers and Applications,* Vol. 27, No. 1 (2005).

[26] Santos, J. and J. Orozco, "Rate monotonic scheduling in hard real-time systems", *Information Processing Letters*, Vol. 48 (1993), pp. 39-45.

[27] Tia, T-S, Z. Deng, M. Shankar, M. Storch, J. Sun, L-C. Wu, J. W-S. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times", *Proceedings 1st IEEE Real-Time Technology and Applications Symposium*, Chicago, Illinois, 1995, pp. 164-173.

[28] Widrow, B.,and S.D. Stearns, Adaptive Signal Processing, Prentice Hall, Englewood Cliff, 1985.

[29] http://www.octave.org

An LP based algorithm for the Privatized Rural Postman Problem

Julián Aráoz
Simón Bolívar University[*], Venezuela
Technical University of Catalonia[†], Spain
julian.araoz@upc.es

Elena Fernández
Technical University of Catalonia, Spain
e.fernandez@upc.es

Oscar Meza[‡]
Simón Bolívar University, Venezuela
meza@ldc.usb.ve

Abstract

In this work we present an LP based algorithm for solving the Privatized Rural Postman Problem. This problem has been defined recently and is a generalization of other arc routing problems like, for instance the Rural Postman Problem. The main difference is that there are no required edges. Instead, there is a profit function on the edges that must be taken into account only the first time that an edge is traversed. The problem is modeled with an integer system of linear inequalities. From the solution of a relaxed model we obtain upper bounds and generate violated cuts when possible. We also propose a heuristic to generate feasible solutions that provide lower bounds. The numerical results from a series of computational experiments with PRPP instances, generated from benchmark RPP instances, are presented.

Keywords: Heuristic Algorithms, Combinatorial Optimization, Arc Routing Problems.

1 Introduction

In this work we present an algorithm to solve the Privatized Rural Postman Problem (PRPP). This problem was defined in Aráoz, Férnandez and Zoltan [3], and is part of the family of Arc Routing Problems (ARPs), that usually aim to determine a least-cost traversal of a specified arc subset of a graph, with or without constraints. Such problems arise in a variety of practical contexts, like post delivery and garbage collection.

Typical constraints require the routes to begin and finish at a given point (the depot), and guarantee the connectivity of the solutions with the depot. We refer the reader to Dror [8] for a recent and comprehensive state on the art of such problems. Here we considered only Edge Routing Problems (ERPs), which are the undirected cases of ARPs.

Like in other ERPs, in the PRPP we assume that the demand for service is placed at the edges of a graph. However, there is no specific arc subset to be traversed. Instead, we assume that giving service to an edge not only would incur a cost (associated with displacement), but would also result on a profit (associated with servicing edges). The displacement cost to an edge will certainly account for the cost of all the edges that are traversed in that same route, as many times as they are traversed. Similarly, the profit associated with servicing an edge, should also take into account the profit of the additional edges that are serviced in that same route. However, the profit of each edge serviced in the route will be collected only once, independently of the number of times the edge is traversed.

[*]Retired Professor Dpt. Process and Systems.
[†]Visiting Professor Dpt. Statistics and Operation Research.
[‡]Retired Professor Dpt. Computation and Information Technology

In the PRPP we look for traversals that maximize the total servicing profit minus the displacement cost. They constitute a generalization of most ERPs and of most Traveling Salesman Problems (TSPs).

The algorithm that we propose for PRPP generates both an upper and a lower bound. The upper bound is derived from solving a series of LP relaxations of an integer system of linear inequalities that define the problem. While possible we reinforce the relaxation by generating cuts violated by the current solution. We use exact separation algorithms for all inequalities.

The lower bound is obtained with a heuristic which is an adaptation of the 3T heuristic of Fernández, Meza, Garfinkel and Ortega [12] for the Rural Postman Problem RPP.

In order to test the efficiency of the proposed algorithm, we have run a series of computational experiments on benchmark instances of classical RPP's where the optimal solution is known. The numerical results assess the good behavior of the method.

The paper is structured as follows. In Section 2 we define the problem and recall from [3] some properties of its solutions. The polyhedral model that we will use is presented in Section 3. Section 4 describes the elements of the algorithm, including the separation procedure for each type of cuts and the heuristic. In Section 5 we describe the computational experiments and present the obtained results. The tables with the numerical results are given in the Appendix. We finish the paper in Section 6 with the conclusions and some remarks about future research.

2 The Privatized Rural Postman Problem

The Privatized Rural Postman Problem was defined in [3]. We next repeat the definitions and results relevant for our algorithm.

Definition 2.1 Privatized Rural Postman Problems (PRPPs).

Given a graph $G(V, E)$ with a distinguished vertex d, the Depot, and two functions from E in $\mathbb{R}_+$, the profit function b and the cost function c, the Privatized Rural Postman Problem is to find one cycle $\mathcal{C}^*$ which maximizes the value of

$$\sum_{e \in \mathcal{C}} (b_e - t_e c_e)$$

where $\mathcal{C}$ is a cycle in G passing trough d, and not necessarily simple, and t_e is the number of times that edge e is traversed in $\mathcal{C}$.

We denote a PRPP by (G, d, b, c).

That is, in a PRPP, we look for the most Profitable Subtour passing through d. In [3] it was proven that the PRPP is NP–Hard.

In this work we model the PRPP by means of a linear system with an exponential number of inequalities, based on the one defined in [3], and we propose a solution approach to obtain lower and upper bounds for the PRPP, that is similar to that in [12] for the RPP. The approach combines the addition of violated cuts to strengthen the LP formulation of the problem, and thus to improve the associated upper bound, with a heuristic to obtain feasible solutions, and thus lower bounds.

We define the functions $\varphi_e = b_e - c_e$ and $\psi_e = b_e - 2c_e = \varphi_e - c_e$. The value φ_e is the net profit obtained when the edge e is traversed once, whereas ψ_e is the the net profit when the edge e is traversed twice.

In the PRPP the edges in $R = \{e \in E \mid \psi_e \geq 0\}$ play an important role, similar to the set of required edges in a RPP (see [3]).

We denote by $G_R \equiv (V(R) \cup \{d\}, R)$ the subgraph induced by the edge set R and the depot. Let $C_i, i \in P = \{0, \ldots, p\}$ be the connected components of the graph G_R and we assume that $d \in C_0$. Using standard graph nomenclature, when necessary, we denote by $V_i = V(C_i)$ and we differentiate between $\gamma(V_i)$ (the set of edges with both ends in V_i) in the original graph G and $\gamma_R(V_i)$ in the graph G_R. That is $\gamma_R(V_i) = \gamma(V_i) \cap R$.

Let e_i^R be an arbitrarily selected edge in $C_i,\ i \in P$, and let $\mathcal{C}^*$ denote an optimal solution. The following properties were proven in [3]:

2.2 *Dominance 1. No edge is used more than twice in $\mathcal{C}^*$.*

2.3 *Dominance 2. If an edge $e \in \mathcal{C}^*$ is used with value c_e then it is used twice in $e \in \mathcal{C}^*$.*

2.4 *Dominance 3. Let $e \in \mathcal{C}^*$, if for some connected component C_k of G_R we have that $V(e) \cap V_k \neq \emptyset$ then all the edges of $\gamma_R(V_k)$ are in $\mathcal{C}^*$. This implies that if $e^1, e^2 \in \gamma_R(V_k)$ then either both edges are in $\mathcal{C}^*$ or none of them is in $\mathcal{C}^*$.*

2.5 *Preprocessing 1. Let C_k be one of the connected components of G_R. If $e \in \gamma(V_k) \setminus R$ then e is used at most once.*

From a more general result in [3] we also have the following corollary:

2.6 *Preprocessing 2. $\gamma_{\mathbf{R}}(\mathbf{V_0}) \subseteq \mathcal{C}^*$.*

Remark 2.7 *By Dominance 1 in any optimal solution to the PRPP no edge is traversed more than twice. Thus, from now on, we will consider an auxiliary graph $G' = (V, E' \cup E'')$, where both E' and E'' are disjoint copies of E with all the edges in E' corresponding to the first time that the original edge is used and all the edges in E'' corresponding to the second time that the original edge is used. We denote by e an original edge in E and by e' and e'' its corresponding copies in E' and E'' respectively.*

Hence for any $e \in E$ we have $b_{e'} = b_e,\ c_{e'} = c_e$ and $b_{e''} = 0,\ c_{e''} = c_e$, also $e_k^R \in E',\ k \in P$. Throughout, e_k^R is chosen to be one of the edges in C_k with greatest φ_e.

3 Polyhedral Model

The linear equations that we use to define the feasible solutions are based in the ones given in the paper of Aráoz, Férnandez and Zoltan [3]. The model is defined on the graph G' of Remark (2.7) as defined above. Each edge $e' \in E'$ is associated with the variable $x_{e'}$ whereas each edge $e'' \in E''$ is associated with the variable $y_{e''}$. Whenever there is not confusion we use x_e and y_e instead of $x_{e'}$ and $y_{e''}$. The PRPP Linear Program (PRPPLP) model is the following:

$$\max z(x,y) = \sum_{e \in E'} \varphi_e x_e - \sum_{e \in E''} c_e y_e \tag{1}$$

$$x(\delta'(v) \setminus F') + y(\delta''(v) \setminus F'') \geq x(F') + y(F'') - |F' \cup F''| + 1,$$
$$v \in V,\ F' \subseteq \delta'(v),\ F'' \subseteq \delta''(v),\ |F' \cup F''| \text{ odd.} \tag{2}$$

$$x(\delta'(S)) + y(\delta''(S)) \geq 2x_e,$$
$$S \subseteq V \setminus \{d\},\ e \in \gamma'(S) \cup \delta'(S),\ V(e) \cap V(R) = \emptyset \tag{3}$$

$$x(\delta'(S)) + y(\delta''(S)) \geq 2x_{e_k^R},$$
$$S \subseteq V \setminus \{d\},\ k \neq 0,\ V_k \cap S \neq \emptyset \tag{4}$$

$$x_e = x_{e_k^R}, \qquad e \in \gamma'(V_k) \cap R',\ k \in P \tag{5}$$

$$y_{e''} \leq x_{e'}, \qquad e \in E \tag{6}$$

$$x_e \leq x_{e_k^R}, \qquad e \in ((\gamma'(V_k) \cup \delta'(V_k)) \setminus R'),\ k \in P \tag{7}$$

$$y_e = 0, \qquad e \in \gamma''(V_k) \setminus R'',\ k \in P \tag{8}$$

$$x_e = 1, \qquad e \in \gamma'(V_0) \cap R' \tag{9}$$

$$x_{e'}, y_{e''} \in \{0,1\}, \qquad e \in E \tag{10}$$

Notice that we use the standard compact notation $f(A) \equiv \sum_{e \in A} f_e$ when $A \subseteq E$, and f is a vector or a function defined on E. Finally, we use the prime and double prime in edge sets to denote which copies we are using, like, for instance, $\delta'(V) = \delta(V) \cap E',\ \delta''(V) = \delta(V) \cap E''$.

- Inequalities (2) ensure even degree at every vertex, and they are implied by the so-called co-circuit inequalities in Barahona and Grötschel [4] in a matroid context (see also Aráoz et al. [1]).
- Inequalities (3) and (4) ensure connectivity with the depot of any edge that is selected, that is any edge e' such that $x_{e'} = 1$. By Dominance 2 we do not need to check the edges in E'' and by Dominance 4 if $e' \in C_k$ we only use e_k^R.
- Equalities (5) correspond to Dominance 4. Inequalities (6) correspond to Dominance 2. Inequalities (7) correspond to Dominance 3 and Dominance 4. Equalities (8) correspond to Preprocessing 1. Equalities (9) correspond to Preprocessing 2. Binary conditions (10) correspond to Dominance 1.

4 Cutting Plane Algorithm

4.1 Initial Linear Relaxation

The cutting plane algorithm starts with the linear relaxation that is described below. At each iteration of the algorithm we add cutting planes, to be described later on. The initial linear relaxation is the following:

$$
\begin{aligned}
&(PRPPLP0)\\
&\max z(x,y) = \sum_{e\in E'} \varphi_e x_e - \sum_{e\in E''} c_e y_e && (11)\\
&x(\delta'(v)) + y(\delta''(v)) \geq 2x_e, \qquad v \in V, e \in \delta'(v). && (12)\\
&x(\delta'(V_k)) + y(\delta''(V_k)) \geq 2x_{e_k^R}, \qquad k \in P \setminus \{0\} && (13)\\
&x_e = x_{e_k^R}, \qquad e \in E' \cap \gamma_R(V_k) \text{ and } k \in P \setminus \{0\} && (14)\\
&y_{e''} \leq x_{e'}, \qquad e \in E && (15)\\
&x_e \leq x_{e_k^R}, \qquad e \in ((\gamma'(V_k) \cup \delta'(V_k)) \setminus R'),\ k \in P && (16)\\
&y_e = 0, \qquad e \in \gamma''(V_k) \setminus R'',\ k \in P && (17)\\
&x_e = 1, \qquad e \in \gamma'(V_0) \cap R' && (18)\\
&0 \leq x_{e'}, y_{e''} \leq 1, \qquad e \in E && (19)
\end{aligned}
$$

This initial program has the same functional (1) and all the inequalities (5)-(9) of PRPPLP. However integrality conditions (10) are relaxed and it only includes those inequalities of the type (2) that correspond to the sets with $|F'| = 1$ and $|F''| = 0$. Note that when $F' = \{e\}$ we have

$$
\begin{aligned}
&x(\delta'(v) \setminus e) + y(\delta''(v)) \geq x_e && \equiv\\
&x(\delta'(v)) - x_e + y(\delta''(v)) \geq x_e && \equiv\\
&x(\delta'(v)) + y(\delta''(v)) \geq 2x_e && \text{(see (12))}
\end{aligned}
$$

No inequality (3) is included but (PRPPLP0) has the inequalities of type (4) that correspond to the sets $S = V_k$ with $e = e_k^R$.

Note that, since some of the original constraints are omitted, the solutions to this system may be integer despite not being feasible to the PRPPLP.

4.2 Cutting Plane Algorithm

The cutting plane algorithm is as follows:

1. Let LPR^0 be the linear program relaxation (11) to (19). Set $k := 0$
2. Let $\bar{z} = \infty$; $\underline{z} = 0$; $optimum = false$; $bestxy = (\overline{0}, \overline{0})$, $end = false$
3. While $end = false$ do:
   a. Find a solution (x^*, y^*) of the current linear program relaxation LPR^k.
   b. if $\bar{z} > z(x^*, y^*)$ then $\bar{z} = z(x^*, y^*)$
   c. If (x^*, y^*) is feasible to PRPPLP then
      $bestxy = (x^*, y^*)$; $end = true$. $\underline{z} = \bar{z}$
      else
      i. Identify inequalities of types (2), (3), (4) violated by (x^*, y^*).
      ii. Add the valid inequalities identified in step (3.c.i) to LPR^k.
      iii. If no new inequality has been added in step (3.c.ii) then
      $end = true$
4. a. Compute the heuristic lower bound and solution $z^h = (x^h, y^h)$

b. if $\underline{z} < z(x^h, y^h)$ then $\underline{z} = z(x^h, y^h)$; $bestxy = (x^h, y^h)$

5. if $\underline{z} = \bar{z}$ then

   output $\underline{z}$; $bestxy$

   else

   output $gap = \bar{z} - \underline{z}$; $\underline{z}$; $bestxy$

4.3 Separation of Cuts

At a given iteration of the algorithm Step 3.*c.i* consists of two parts: first we find violated inequalities of type (2) and second we find violated inequalities of type (3) or (4), if it is possible.

4.3.1 Finding violated inequalities of type (2)

To separate the inequalities (2) we use a modification of a heuristic proposed by Ghiani and Laporte [13] for the RPP. This modification results in an exact separation method for singletons although the separation is still a heuristic for general vertex sets.

Given a graph G and a vector $x : E \to [0, 1]$, the cocycle inequalities for vertex v are:

$$x(\delta(v) \setminus F) \geq x(F) - \mid F \mid +1, \ \forall F \subseteq \delta(v), \ F \text{ odd} \tag{20}$$

The separation problem for these inequalities is: Given an vector x^*, $0 \leq x^* \leq 1$ determine if there is one of the inequalities violated by this vector or determine that none exists.

4.1 Algorithm:

1. For each vertex v do

   (a) Let $F = \{e \in \delta(v) | x^*_e \geq 0.5\}$.

   (b) If F is even then let $x^*_{e^1} = \min\{x^*_e | e \in F\}$ and $x^*_{e^2} = \max\{x^*_e | e \in \delta(v) \setminus F\}$. If $x^*_{e^1} - 0.5 \leq 0.5 - x^*_{e^2}$ delete e^1 from F otherwise add e^2 to F.

   (c) If the Inequality (20) is violated by this set F we have a cut.

Theorem 4.2 The separation Algorithm (4.1) is exact and the order of the algorithm is $|E|$.
Proof: Consider rewriting the inequalities (20):

$$x^*(\delta(v) \setminus F) + \sum_{e \in F} (1 - x^*_e) \geq 1, \ |F| \text{ odd}. \tag{21}$$

It is clear that an edge e contributes to the left hand side of the above expression with value $(1 - x^*_e)$ if it is in F and with value x^*_e if it is not. Therefore, Step (1a) gives the set F with minimum value of the left hand side and Step (1b) gives the minimum correction to get an odd set.

Hence, Algorithm 4.1 is exact. □

Let (x^*, y^*) be a solution of the current linear program relaxation LPR. Applying Algorithm (4.1) to this solution we obtain an odd set $(F' \cup F'')$. If

$$x^*(\delta'(v) \setminus F') + y^*(\delta''(v) \setminus F'') + \sum_{e \in F'} (1 - x^*_e) + \sum_{e \in F''} (1 - y^*_e) - 1 < 0$$

then the following inequality is violated by (x^*, y^*):

$$x(\delta'(v) \setminus F') + y(\delta''(v) \setminus F'') \geq x(F') + y(F'') - |F' \cup F''| + 1$$

4.3.2 Finding violated inequalities of type (3) or (4)

Inequalities of type (3 can be classified in two types. The inequalities of the first type are of the form $x(\delta(S)) + y(\delta(S)) \geq 2x_e$, for $e \in \delta(S)$, $S \subseteq V \setminus \{d\}$. We will call thenm set-parity inequalities (as opposed to inequalities (2) which are referred to as node-parity inequalities).

The other type of inequalities (3) are of the form $x(\delta(S)) + y(\delta(S)) \geq 2x_e$, for $e \in \gamma(S)$, $S \subseteq V \setminus \{d\}$. We call them connectivity inequalities.

The inequalities (4) can also be classified in these two classes: set-parity inequalities ($x_{e_k^R} \in \delta(S)$) and connectivity inequalities ($x_{e_k^R} \notin \delta(S)$).

For separating any of the above inequalities (connectivity or set-parity) we use the algorithm of Gusfield [14] to calculate the tree of min-cuts of the graph G'', obtained from the graph G' by eliminating all edges in E' or E'' with value zero in the current solution (x^*, y^*), and by assigning a cost to each edge with value equal to the value of the variable associated with that edge.

The connectivity inequalities are separated with the exact algorithm of Belenguer and Benavent [5, 6]. The algorithm does the following:

For each edge $e = \{u, v\} \in E'$ with v and w different from the depot and $x_e^* > 0$ the minimum cut $\delta'(S) \cup \delta''(S)$ such that $e \in \gamma'(S)$ is easily obtained from the min-cut tree. If the value of the min cut is smaller than $2x_e^*$ then the following inequality is violated by (x^*, y^*): $x(\delta'(S)) + y(\delta''(S)) \geq 2x_{e'}$.

The set parity inequalities are separated exactly with a very simple algorithm: For each edge $e = \{u, v\} \in E'$ we determine the minimum cut $\delta'(S) \cup \delta''(S)$ that separates v and u. If the value of this min cut is less than $2x_e$ then the following inequality is violated by (x^*, y^*): $x(\delta'(S)) + y(\delta''(S)) \geq 2x_e$.

4.4 Heuristic Bound

The algorithm used to obtain a lower bound is the following heuristic:

1. Let (x^*, y^*) be the solution obtained with the linear relaxation described in the previous section.
2. Transform the PRPP into a RPP. The graph of the RPP will be the original graph G of PRPP plus a fictitious edge $\{1, 1'\}$. Edge $\{1, 1'\}$ is defined as required to guarantee that the solution passes through the depot. All other edges of PRPP with value x_e^* greater or equal to ϵ are also defined as required edges for RPP (ϵ is a parameter). The cost of edge $\{1, 1'\}$ is zero, and the cost of any other edge is the same as the costs of the edge in PRPP.
3. Find a feasible solution to RPP with the 3T heuristic of Fernández, Meza, Garfinkel and Ortega [12].
4. The feasible solution to PRPP results from eliminating the parallel edges $\{1, 1'\}$ from the feasible solution to RPP obtained in (3).

5 Computational Experiments

In order to evaluate the performance of the proposed algorithm we have run a series of computational experiments (see Aráoz, Férnandez and Meza[2]). We next describe some of these experiments and give the obtained numerical results. Programs have been coded in C using CPLEX 7.0 library. All instances were run on a Sun ULTRA 10, model 440, 1x440 MHZ, 1GB DRAM. In this section use the Acronyms in Table 1 for the identification of columns in the tables of results. Since there are no available benchmark instances for PRPP, we have generated PRPP instances from well known sets of RPP benchmark instances.

The RPP benchmark instances are divided in five groups. The first group contains two problems, ALBAIDAA and ALBAIDAB, obtained from the Albaida, Spain Graph (see Corberán and Sanchis [10]). The second group contains the 24 instances (problems labeled P) of Christofides et al. [9]. The last three groups contain instances from Hertz et al [15]: 36 instances with vertices of degree 4 and disconnected required edge sets (labeled DEGREE), 36 grid instances (labeled GRID), and 20 randomly generated instances (labeled RANDOM).

The set of PRPP instances was obtained in such a way that the optimal solutions were known, so we could better evaluate the obtained results. From [3] we know that this can be achieved by keeping in PRPP the cost function (c) of RPP and by defining a profit function (b) in PRPP that assigns high enough profits

to the edges that are required in RPP and zero profit for the edges that are non-required in RPP. Along this line, the set of test problems was generated by taking $b_e = 1000 * c_e, \forall e \in E_R$, and $b_e = 0, \forall e \in E \setminus E_R$.

The results with this set of instances are given in Tables 2 and 3. As can be seen the obtained results are very good. From the optimal/best known solutions to the RPP instances, we know when the solutions obtained by the heuristic are/are not optimal for the PRPP. In particular, the heuristic provided the optimal solution for 81 of the 118 instances. The upper bound of the percent gap between the optimal and heuristic values (PGHO column) for the non optimal solutions does not exceed 0.01% for all instances, and, except for instance DEGREE1, the corresponding percent gap between costs (PGHOC column) is less than 4%. We can conclude that both the upper and lower bounds were very good for these instances. In general, the number of LP iterations of the algorithm is small. For 92 instances this number does not exceed 20, and only 2 instances required more than 100 iterations to terminate. In general, the instances that require more iterations are the ones with a larger number of connected componentes in the graph induced by the requires edges of RPP. These are the instances that also tend to be harder to solve as RPP instances. To some extent the difficulty for solving the instances is also reflected on the number of cuts that were added: both in the total number of cuts and on the number of cuts per iteration. In the vast majority of the cases there are more connectivity cuts than parity cuts (either set parity or node parity). However, in our opinion this number is small in all the cases taking into account the difficulty of the problems. The efficiency of the algorithm can also be appreciated in the required cpu times. As was expected, most of the cpu time is consumed for obtaining the lower bound. There are four instances that required more than 200 seconds, although in general these times were small, and 70 instances required less than one second. The times of the heuristic are very small: they only exceed one second in 14 instances and the largest time is below 10 seconds.

6 Conclusions

In this work we have presented an algorithm to obtain upper and lower bounds for the Privatized Rural Postman Problem that was introduced in [3]. The lower bound is obtained with an iterative LP-based cutting plane algorithm and the upper bound is obtained with a heuristic that takes as starting point the solution of the last LP relaxation. We have presented procedures to solve exactly the separation problem for the generated cuts. In order to evaluate the performance of the algorithm we have generated a set of PRPP instances from a set of benchmark RPP instances.

Acknowledgements

The work of the first and third authors has been partially supported by grants from FONACIT Project USB S1- 2000000438, Venezuela. The research of the first author has been partially financed by the Spain "Secretaría de Estado de Educación y Universidades". The research of the second author has been partially supported by grant TIC 2003-05982-C05-04 of the Inter-Ministerial Spanish Commission of Science and Technology. These supports are gratefully acknowledged.

References

[1] J. Aráoz , W. Cunningham , J. Edmonds, and J. Green–Krotki. Reductions to 1–matching polyhedra. *Networks*, 13:455–473, 1983.

[2] J. Aráoz, E. Fernández and O. Meza. "An LP based algorithm for the Privatized Rural Postman Problem". Reporte DR-2005/06, Departament Estadística I Investigació Operativa, Universitat Politècnica de Catalunya, España (2005).

[3] J. Aráoz, E. Fernández, C. Zoltan. "The Privatizad Rural Postman Problem". Reporte DR-2003/12, EIO Departament , Universitat Politcnica de Catalunya, Espaa (2002). Accepted in *Computers and Operations Research*.

[4] F. Barahona and M. Groeschel. On the cycle polytope of a binary matroid.*J. Comb. Theory*, 40:40–62, 1986.

[5] J. M. Belenguer and E. Benavent. The capacitated arc routing problem: valid inequalities and facets. *Computational Optimization and Applications* 10, 165-187, 1998.

[6] E. Benavent, A. Corberan, and J.M. Sanchis. Linear programming based methods for solving arc routing problems. In *Arc Routing: Theory, Solutions and Applications* M. Dror (edt), Kluwer Academic Publishers, 2000.

[7] C. Berge *Graphs and Hypergraphs*, North-Holland, Amsterdam, 1973.

[8] M. Dror Edt. *Arc Routing: Theory, Solutions and Applications*. Kluwer Academic Publishers, 2000.

[9] N. Christofides, V. Campos, A. Corberan, and E. Mota. An algorithm for the rural postman problem. *Imperial College Report IC.O.R.*, 81.5, 1981.

[10] A. Corberán, J. M. Sanchis. A polyhedral approach to the rural postman problem. *Eur. J. Oper. Res.* 79, 95-114, 1994.

[11] A. Corberán, J. M. Sanchis. The General Routing Problem: facets from the RPP and GTSP polyhedra. *Eur. J. Oper. Res.* 108, 538-550, 1998.

[12] E. Fernández, O. Meza, R. Garfinkel, and M. Ortega. On the undirected rural postman problem: Tight bounds based on a new formulation. *Operations Research*, 51:281–291, 2003.

[13] G. Ghiani and G. Laporte. A branch-and-cut algorithm for the undirected rural postman problem. *Mathematical Programming*, 87:467–481, 2000.

[14] D. Gusfield. Very Simple Methods for All Pairs Network Flow Analysis. *SIAM Journal of Computing* 19, 143-155, 1990.

[15] A. G. Hertz, P. Laporte, H. Nanchen. Improvement procedures for the undirected rural postman problem. *INFORMS J. Comput.* 1, 53-62, 1999.

APPENDIX

In this Appendix we concentrate the tables showing the results of our experiments, the analysis of these results are in Section (5).

Acronyms	Parameter Name
PN	Problem Name
NV	Number of Vertices
NE	Number of Edges
NCR	Number of Connected components in the graph induced by the Required edges
LRV	Linear Relaxation Value
HV	Heuristic Value
OV	Optimal Value
PGHO	Upper Bound of the Percent Gap between HV and OV = 100*(OV-HV)/OV (or 100*(LRV-HV)/LRV, if OV is not defined)
TPHS	Total Profit of the Heuristic Solution
TCHS	Total Cost of the Heuristic Solution
TCOS	Total Cost of the Optimal Solution
PGHOC	Upper Bound of the Percent Gap between Optimal and Heuristic Solution Costs
LRT	Linear Relaxation Time (sec)
HT	Heuristic Time (sec)
NLR	Number of Linear Relaxations solved
NCLR	Number of Connectivity Constraints added to Linear Relaxation
NSPLR	Number of Set Parity Constraints added to Linear Relaxation
RVPRN	Number of Vertex Parity Constraints added to Linear Relaxation

Table 1: Table Column Acronyms

PN	NV	NE	NRC	LRV	HV	OV	PGHO	TPHS	TCHS	TCOS	PGHOC	LRT	HT	NLR	NCLR	NSPLR	RVPEN
ALBAIDAA	102	160	10	7284557	7284401	7284401	0,000	7295000	10599	10599	0,00	4,51	1,50	6	126	23	38
ALBAIDAB	90	144	11	5794551	5794371	5794371	0,000	5803000	8629	8629	0,00	2,60	1,24	6	227	22	38
P01	11	13	4	24924	24924	24924	0,000	25000	76	76	0,00	0,01	0,17	3	11	1	2
P02	14	33	4	79858	79848	79848	0,000	80000	152	152	0,00	0,02	0,04	2	9	2	2
P03	28	58	4	69899	69898	69898	0,000	70000	102	102	0,00	0,11	0,06	5	24	1	4
P04	17	35	3	54919	54916	54916	0,000	55000	84	84	0,00	0,03	0,02	3	34	1	7
P05	20	35	5	68880	68876	68876	0,000	69000	124	124	0,00	0,04	0,06	3	11	4	3
P06	24	46	7	69900	69898	69898	0,000	70000	102	102	0,00	0,06	0,03	3	19	6	7
P07	23	47	3	92870	92870	92870	0,000	93000	130	130	0,00	0,04	0,01	3	29	1	10
P08	17	40	2	92879	92878	92878	0,000	93000	122	122	0,00	0,06	0,02	5	0	0	13
P09	14	26	3	56919	56917	56917	0,000	57000	83	83	0,00	0,18	0,01	4	9	2	6
P10	12	20	4	44928	44920	44920	0,000	45000	80	80	0,00	0,01	0,03	2	7	2	0
P11	9	14	3	13977	13977	13977	0,000	14000	23	23	0,00	0,01	0,00	1	0	0	0
P12	7	18	3	12981	12981	12981	0,000	13000	19	19	0,00	0,33	0,00	2	4	0	0
P13	7	10	3	11965	11965	11965	0,000	12000	35	35	0,00	0,01	0,01	1	0	0	0
P14	28	79	6	144812	144798	144798	0,000	145000	202	202	0,00	0,06	0,19	2	28	4	11
P15	26	37	8	179562	179559	179559	0,000	180000	441	441	0,00	0,07	0,18	4	56	8	6
P16	31	94	7	138802	138797	138797	0,000	139000	203	203	0,00	0,17	0,54	4	64	5	8
P17	19	44	5	62893	62888	62888	0,000	63000	112	112	0,00	0,03	0,08	2	17	4	2
P18	23	37	8	60856	60854	60854	0,000	61000	146	146	0,00	0,05	0,05	3	38	6	8
P19	33	55	7	140744	140743	140743	0,000	141000	257	257	0,00	0,11	0,06	3	31	1	8
P20	50	98	7	281615	281596	281602	0,002	282000	404	398	1,51	0,51	3,29	6	100	1	30
P21	49	110	6	287643	287630	*	0,005	288000	370	*	3,51	0,39	0,69	5	172	9	23
P22	50	184	6	498385	498379	498379	0,000	499000	621	621	0,00	0,44	0,10	4	131	0	27
P23	50	158	6	379528	379525	379525	0,000	380000	475	475	0,00	0,65	1,05	6	208	0	38
P24	41	125	7	291598	291595	291595	0,000	292000	405	405	0,00	0,26	0,11	4	112	1	17
DEGREE0	16	32	2	118728	118728	118728	0,000	119000	272	272	0,00	0,02	0,00	5	10	1	0
DEGREE1	16	31	3	246306	246227	246299	0,029	247000	773	701	10,27	0,07	0,03	11	76	21	1
DEGREE2	16	31	4	321351	321298	321298	0,000	322000	702	702	0,00	0,02	0,01	2	9	4	1
DEGREE3	16	32	3	342246	342246	342246	0,000	343000	754	754	0,00	0,08	0,00	12	72	36	2
DEGREE4	16	31	4	335159	335080	335080	0,000	336000	920	920	0,00	0,06	0,03	8	60	14	0
DEGREE5	16	31	4	489106	489037	489037	0,000	490000	963	963	0,00	0,03	0,05	4	18	3	4
DEGREE6	16	32	5	404148	404091	404091	0,000	405000	909	909	0,00	0,03	0,03	4	40	7	2
DEGREE7	16	31	4	463064	462966	462974	0,002	464000	1034	1026	0,78	0,03	0,12	3	29	3	6
DEGREE8	16	31	4	613071	612987	612987	0,000	614000	1013	1013	0,00	0,03	0,12	3	12	4	6
DEGREE9	36	72	8	364986	364963	364968	0,001	366000	1037	1032	0,48	0,21	0,05	8	130	50	3
DEGREE10	36	72	7	234261	234261	234261	0,000	235000	739	739	0,00	1,23	0,02	26	559	168	2
DEGREE11	36	72	11	627912	627912	627912	0,000	629000	1088	1088	0,00	0,19	0,08	5	117	35	1
DEGREE12	36	72	8	512915	512915	512915	0,000	514000	1085	1085	0,00	0,16	0,03	5	86	46	6
DEGREE13	36	72	5	723896	723896	723903	0,001	725000	1104	1097	0,64	0,12	0,10	3	54	18	2
DEGREE14	36	72	9	956609	956584	956584	0,000	958000	1416	1416	0,00	0,15	0,03	3	32	11	8
DEGREE15	36	72	6	911619	911595	911595	0,000	913000	1405	1405	0,00	0,15	0,08	4	33	9	11
DEGREE16	36	72	4	1064621	1064553	1064564	0,001	1066000	1447	1436	0,77	0,20	0,09	4	77	17	14
DEGREE17	36	72	6	1233355	1233243	1233317	0,006	1235000	1757	1683	4,40	0,33	0,11	5	6	2	16
DEGREE18	64	128	15	715750	715685	715736	0,007	717000	1315	1264	4,03	23,03	0,63	54	1654	252	3
DEGREE19	64	128	11	727649	727607	727627	0,003	729000	1393	1373	1,46	25,51	0,21	15	359	140	7
DEGREE20	64	128	10	660757	660737	660738	0,000	662000	1263	1262	0,08	3,66	0,09	15	773	155	11
DEGREE21	64	128	10	1149416	1149386	1149398	0,001	1151000	1614	1602	0,75	1,41	0,30	6	213	43	23
DEGREE22	64	128	12	1130332	1130245	*	0,008	1132000	1755	*	4,96	1,23	0,46	5	183	55	17
DEGREE23	64	128	9	1172281	1172226	1172226	0,000	1174000	1774	1774	0,00	1,61	0,14	8	184	43	27
DEGREE24	64	128	6	1720842	1720805	1720812	0,0004	1723000	2195	2188	0,32	0,98	0,12	5	52	12	31
DEGREE25	64	128	9	1472045	1471986	1472011	0,002	1474000	2014	1989	1,26	1,15	0,55	6	125	22	32
DEGREE26	64	128	5	1865634	1865532	1865559	0,001	1868000	2468	2441	1,11	0,93	0,27	5	135	21	42
DEGREE27	100	200	22	901275	901207	901207	0,000	903000	1793	1793	0,00	58,50	1,95	28	1600	377	17
DEGREE28	100	200	19	1036196	1036118	1036166	0,005	1038000	1882	1834	2,62	201,50	3,00	73	5407	706	21
DEGREE29	100	200	20	900310	900194	*	0,013	902000	1806	*	6,42	304,66	6,80	134	8757	1388	19
DEGREE30	100	200	13	1549755	1549690	1549690	0,000	1552000	2310	2310	0,00	18,37	0,81	17	1669	140	43
DEGREE31	100	200	10	1781503	1781461	1781461	0,000	1784000	2539	2539	0,00	42,05	1,65	31	2732	159	51
DEGREE32	100	200	16	1475919	1475836	1475863	0,002	1478000	2164	2137	1,26	4,43	4,04	5	244	90	26
DEGREE33	100	200	9	2159194	2159054	*	0,006	2162000	2946	*	4,75	7,83	10,29	7	393	29	53
DEGREE34	100	200	9	2204135	2204068	2204094	0,001	2207000	2932	2906	0,89	5,26	0,31	7	622	34	52
DEGREE35	100	200	9	2179269	2179154	2179180	0,001	2182000	2846	2820	0,92	4,66	0,55	5	231	29	55

(*) The best known OV values for problems P21, DEGREE 22, 29 and 33 are respectively 287630, 1130260, 900194, 2159072

Table 2: Experiments RPP

PN	NV	NE	NRC	LRV	HV	OV	PGHO	TPHS	TCHS	TCOS	PGHOC	LRT	HT	NLR	NCLR	NSPLR	RVPRN
GRID0	16	24	3	2988	2988	2988	0,000	3000	12	12	0,00	0,05	0,01	8	80	18	0
GRID1	16	24	5	4986	4986	4986	0,000	5000	14	14	0,00	0,05	0,02	6	30	11	0
GRID2	16	24	4	3988	3988	3988	0,000	4000	12	12	0,00	0,20	0,01	8	54	11	0
GRID3	16	24	5	7984	7984	7984	0,000	8000	16	16	0,00	0,02	0,01	2	7	8	0
GRID4	16	24	5	6984	6984	6984	0,000	7000	16	16	0,00	0,05	0,02	5	54	10	1
GRID5	16	24	3	6987	6986	6986	0,000	7000	14	14	0,00	0,19	0,02	7	58	14	1
GRID6	16	24	4	12981	12980	12980	0,000	13000	20	20	0,00	0,03	0,06	3	10	5	1
GRID7	16	24	4	7984	7984	7984	0,000	8000	16	16	0,00	0,03	0,01	3	24	9	1
GRID8	16	24	4	8984	8982	8982	0,000	9000	18	18	0,00	0,05	0,05	5	50	11	2
GRID9	36	60	7	10976	10976	10976	0,000	11000	24	24	0,00	1,83	0,02	28	511	221	3
GRID10	36	60	9	12971	12970	12970	0,000	13000	30	30	0,00	0,88	0,06	14	408	66	2
GRID11	36	60	9	14971	14970	14970	0,000	15000	30	30	0,00	2,39	0,12	16	243	111	4
GRID12	36	60	7	25959	25956	25958	0,008	26000	44	42	4,76	0,47	0,42	8	246	13	11
GRID13	36	60	6	22962	22960	22960	0,000	23000	40	40	0,00	0,78	0,10	16	347	65	9
GRID14	36	60	7	24960	24958	24958	0,000	25000	42	42	0,00	0,36	0,35	6	138	21	8
GRID15	36	60	5	34953	34950	34952	0,006	35000	50	48	4,17	0,12	0,22	3	25	7	10
GRID16	36	60	7	29957	29954	29954	0,000	30000	46	46	0,00	0,16	0,28	4	113	10	6
GRID17	36	60	5	33952	33952	33952	0,000	34000	48	48	0,00	0,12	0,03	3	67	6	11
GRID18	64	112	10	23953	23950	23952	0,008	24000	50	48	4,17	15,53	0,21	35	1797	227	7
GRID19	64	112	12	26950	26948	26948	0,000	27000	52	52	0,00	35,01	0,24	70	2939	312	5
GRID20	64	112	14	26948	26946	26946	0,000	27000	54	54	0,00	39,88	0,15	72	2269	383	7
GRID21	64	112	8	45933	45930	45930	0,000	46000	70	70	0,00	17,85	0,27	34	1753	130	20
GRID22	64	112	8	46933	46932	46932	0,004	47000	70	68	2,94	2,21	0,26	8	401	37	13
GRID23	64	112	11	49926	49924	49926	0,004	50000	76	74	2,70	0,95	0,41	5	250	33	16
GRID24	64	112	4	67911	67910	67910	0,000	68000	90	90	0,00	1,73	0,22	8	75	14	27
GRID25	64	112	5	60917	60916	60916	0,000	61000	84	84	0,00	1,33	0,22	5	148	13	27
GRID26	64	112	6	65917	65916	65916	0,000	66000	84	84	0,00	1,07	0,13	5	225	12	21
GRID27	100	180	19	40923	40920	*	0,007	41000	80	*	3,75	144,78	0,72	71	3504	521	16
GRID28	100	180	20	48916	48914	48914	0,000	49000	86	86	0,00	207,10	0,57	87	7680	438	25
GRID29	100	180	20	43920	43916	*	0,009	44000	84	*	4,76	370,23	1,53	192	8563	1208	18
GRID30	100	180	13	72893	72890	72890	0,000	73000	110	110	0,00	27,05	0,73	21	1730	270	31
GRID31	100	180	18	76888	76884	76886	0,003	77000	116	114	1,75	7,21	7,20	7	438	100	19
GRID32	100	180	11	81883	81878	*	0,006	82000	122	*	4,10	3,73	5,21	6	51	23	29
GRID33	100	180	4	112854	112852	112852	0,000	113000	148	148	0,00	3,57	0,34	4	62	6	39
GRID34	100	180	9	106859	106856	106856	0,000	107000	144	144	0,00	7,31	2,05	10	150	9	41
GRID35	100	180	6	108860	108858	108858	0,002	109000	144	142	1,41	4,81	2,35	6	162	14	44
RANDOM0	20	37	3	6219147	6219147	6219147	0,000	6249000	29853	29853	0,00	0,16	0,01	16	154	30	1
RANDOM1	20	47	4	9687683	9687683	9687683	0,000	9722000	34317	34317	0,00	0,17	0,00	15	128	35	0
RANDOM2	20	47	3	8368032	8368032	8368032	0,000	8393000	24968	24968	0,00	0,23	0,01	20	126	76	1
RANDOM3	20	75	4	17443946	17443946	17443946	0,000	17478000	34054	34054	0,00	0,09	0,00	8	61	24	4
RANDOM4	20	60	3	16749923	16749744	16749744	0,000	16776000	26256	26256	0,00	0,14	0,01	11	98	22	2
RANDOM5	30	70	4	9271242	9271242	9271242	0,000	9307000	35758	35758	0,00	0,72	0,01	23	517	84	5
RANDOM6	30	111	4	21471266	21470779	21470779	0,001	21510000	39358	39221	0,35	1,30	0,00	42	495	108	2
RANDOM7	30	70	5	14375860	14375860	14375860	0,000	14411000	35140	35140	0,00	0,50	0,01	26	163	95	1
RANDOM8	30	111	5	22854271	22854271	22854271	0,000	22903000	48729	48729	0,00	1,75	0,01	42	784	149	2
RANDOM9	30	111	6	22464428	22464428	22464428	0,000	22507000	42572	42572	0,00	0,29	0,01	11	148	70	4
RANDOM10	40	130	8	20616259	20616259	20616259	0,000	20655000	38741	38741	0,00	2,76	0,02	30	530	154	1
RANDOM11	40	103	6	15172873	15172873	15172372	0,000	15215000	42628	42628	0,00	1,25	0,02	31	473	129	4
RANDOM12	40	82	5	11295655	11294601	11295414	0,007	11346000	51399	50586	1,61	1,74	0,05	32	809	87	3
RANDOM13	40	203	9	35106002	35105133	35105133	0,000	35160000	54867	54867	0,00	0,47	0,05	9	94	32	3
RANDOM14	40	203	7	41966744	41965058	41965451	0,001	42031000	65942	65549	0,60	1,29	0,13	16	305	124	4
RANDOM15	50	203	8	36103911	36101588	36102286	0,002	36166000	64412	63714	1,10	12,14	0,10	53	1765	124	10
RANDOM16	50	162	12	19338651	19338651	19338651	0,000	19389000	50349	50349	0,00	2,29	0,06	29	440	233	2
RANDOM17	50	130	6	15917728	15917728	15917728	0,000	15961000	43272	43272	0,00	5,16	0,11	49	1199	305	3
RANDOM18	50	203	7	35604130	35604130	35604130	0,000	35661000	56870	56870	0,00	5,57	0,01	38	1129	160	9
RANDOM19	50	203	8	31646637	31646637	31645669	0,001	31691000	45661	45331	0,73	0,37	0,04	5	66	42	2

(*) The best known OV values for problems GRID 27, 29 and 32 are respectively 40920, 43916, 81880

Table 3: Experiments RPP (continuation)

Grasp en la Resolución del Problema de Clustering

Erick V. Vicente
Universidad Nacional Mayor de San marcos, Unidad de Postgrado FISI,
Lima, Perú, Lima 01
erick.vicente@gmail.com

Luis A. Rivera
Universidad Federal Norte Fluminense, LCMAT-CCT,
Campos dos Goytacazes, Rio de Janeiro, Brasil, 28015-620
rivera@uenf.br

David S. Mauricio
Universidad Ricardo Palma, Facultad de Ingeniería,
Lima, Perú, Lima 33
Universidad Nacional Mayor de San marcos, Unidad de Postgrado FISI,
Lima, Perú, Lima 01
dms_research@yahoo.com

Abstract

The clustering could be approached as a combinatorial optimization problem when the clusters are a partition of an objects set. The Grasp meta-heuristic is a relatively recent technic that had been used to solve of an efficient manner several combinatorial optimization problems. In this work, we adapted the Grasp meta-heuristic to solve the clustering problem based on the basis of K-Means algorithm. The proposed algorithm, named GraspKM, takes advantages of fast convergence of K-Means algorithm avoiding the inconvenience of obtaining a local optimal. The algorithm shows to be better than K-Means algorithm and it is comparable with another meta-heuristic method reviewed with respect to efficiency. The computational experiments had been realized with a data collection extensively used on clustering literature.

Keywords: Grasp, K-Means, Clustering, Classification, Meta Heuristic.

Resumen

El clustering puede ser abordado como un problema de optimización combinatoria cuando los clusters son una partición de un conjunto de objetos. La metaheurística Grasp es una técnica relativamente reciente que ha sido utilizada para resolver de manera eficiente múltiples problemas de optimización combinatoria. En este trabajo, adaptamos la metaheurística Grasp para la resolución del problema del clustering basado en los principios del algoritmo K-Means. El algoritmo propuesto, denominado GraspKM, aprovecha la rápida convergencia del algoritmo K-Means evitando el inconveniente de alcanzar óptimos locales. El algoritmo demuestra ser superior al algoritmo K-Means y es comparable con otras metaheurísticas revisadas en cuanto a eficiencia. Los experimentos computacionales han sido realizados con colecciones de datos ampliamente usados en la literatura sobre clustering.

Palabras claves: Grasp, K-Means, Clustering, Clasificación, Meta Heurística.

1. INTRODUCCIÓN

Dado un conjunto de objetos, el proceso de clustering debe encontrar grupos cuyos elementos sean similares entre sí y a la vez diferentes a los elementos de los otros grupos. Los grupos con esas características

son conocidos como *clusters*. Los objetos son representados por D atributos descriptores en forma de vectores en el espacio R^D, y con una medida de comparación de la similaridad, como la distancia, se conformarán los clusters con objetos similares. En el proceso de la conformación de los grupos, que en adelante será conocido como clustering, no existe conocimiento previo acerca de cómo se debe conformar un cluster; por tal motivo, el proceso de clustering es también conocido como clasificación no supervisada. Los tipos de objetos varían de acuerdo al contexto de la aplicación del clustering; por ejemplo, en las tareas de clasificación dentro de la minería de datos, los objetos serán registros de la base de datos; en la recuperación de la información los objetos serán documentos; y en procesamiento de imágenes los objetos serán los píxeles que conforman la imagen.

El clustering tiene múltiples aplicaciones dentro de las ciencias de la computación, como compresión de imágenes [25] y voz digitalizadas [17]; en la recuperación de informaciones relacionadas [2]; en minería de datos, donde se buscan grupos con ciertas características de interés (por ejemplo, descubrimiento de nuevos segmentos de clientes con el fin de mejorar los servicios que brinda una determinada empresa) [8]; en la segmentación de imágenes para dividir la imagen en regiones homogéneas (según alguna característica de interés como la intensidad, color o textura) en aplicaciones médicas [24], clasificación de imágenes satelitales en zonas (urbana, descampados, bosques, ríos) [26].

Los métodos de clustering existentes difieren uno del otro en la forma de estructurar los clusters. Aquellos que encuentran clusters que corresponden a una partición del conjunto de objetos se les conoce como métodos de Hard-clustering [16] o clustering particional [13], siendo el más conocido el algoritmo K-Means [10, 18]. Los métodos que asignan a cada objeto un valor de pertenencia con respecto a cada cluster se les conoce como métodos de Soft-clustering [16], y entre los más representativos de este tipo de clustering se encuentran los algoritmos Fuzzy C-Means [4] y Expectation Maximization [7]. El método propuesto en el presente trabajo se considera dentro de las técnicas de Hard-clustering o clustering particional; por tanto, se definirá el problema del clustering desde ese punto de vista.

Problema de Clustering

Dado un conjunto de n objetos denotado por $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, en que $\boldsymbol{x}_i \in \boldsymbol{R}^D$. Sea K un número entero positivo conocido a priori, el problema del clustering consiste en encontrar una partición $\boldsymbol{P} = \{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_K\}$ de $\boldsymbol{X}$, siendo $\boldsymbol{C}_j$ un cluster conformado por objetos similares, satisfaciendo una función objetivo $f : \boldsymbol{R}^D \rightarrow \boldsymbol{R}$, y las condiciones:

$$\boldsymbol{C}_i \bigcap \boldsymbol{C}_j = \emptyset \text{ para } i \neq j, \quad \text{y} \quad \bigcup \boldsymbol{C}_i = \boldsymbol{X}.$$

Para medir la similaridad entre dos objetos $\boldsymbol{x}_a$ y $\boldsymbol{x}_b$ se usará una función de distancia denotada por $d(\boldsymbol{x}_a, \boldsymbol{x}_b)$, siendo la distancia Euclidiana la más usada para medir la similaridad. Así la distancia entre dos diferentes elementos $\boldsymbol{x}_i = (x_{i1}, ...x_{iD})$ y $\boldsymbol{x}_j = (x_{j1}, ..., x_{jD})$ es $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{l=1}^{D}(x_{il} - x_{jl})^2}$. Los objetos de un cluster son similares cuando las distancias entre ellos es mínima; esto permite formular la función objetivo f, como:

$$\sum_{j=1}^{K} \sum_{\boldsymbol{x}_i \in \boldsymbol{C}_j} d(\boldsymbol{x}_i, \bar{\boldsymbol{x}}_j)^2; \tag{1}$$

esto es, se desea minimizar (1); donde $\bar{\boldsymbol{x}}_j$, conocido como elemento representativo del cluster, es la media de los elementos del cluster $\boldsymbol{C}_j$,

$$\bar{\boldsymbol{x}}_j = \frac{1}{|\boldsymbol{C}_j|} \sum \boldsymbol{x}_i \in \boldsymbol{C}_j, \tag{2}$$

y corresponde al centro del cluster.

Bajo esas características, el clustering es un problema de optimización combinatoria, y ha sido demostrado que es un NP-Difícil [5].

Método Propuesto

El algoritmo K-Means es una técnica clustering ampliamente usada y capaz de encontrar rápidamente una solución minimizando (1). Dentro de los principales inconvenientes del algoritmo K-Means está su alta dependencia de la elección de los centros iniciales y su convergencia a óptimos locales. Esta última

deficiencia es en realidad una debilidad de los algoritmos golosos, que son rápidos encontrando soluciones pero quedan atrapados en óptimos locales. Feo y Resende [9] proponen una metaheurística llamada Grasp (Greedy Randomised Adaptative Search Procedure) que aprovecha la efectividad de los algoritmos golosos adaptando la forma voraz de construir las soluciones para evitar la convergencia a óptimos locales y luego mejorar las soluciones encontradas.

El presente trabajo tiene como objetivo la adaptación del algoritmo K-Means dentro del procedimiento Grasp para la obtención de mejores soluciones que el algoritmo K-Means y comparables a las soluciones obtenidas con otras metaheurísticas propuestas para el problema del clustering. Esto es, respecto a inestabilidad de los resultados y robustez del método.

Para alcanzar el objetivo propuesto en este trabajo, en la Sección 2 se hace una breve revisión de los métodos de clustering. En la Sección 3 se presenta el algoritmo GraspKM para el problema del clustering. En la Sección 4 se describe las colecciones de datos usadas para la prueba del algoritmo, así como el análisis de los resultados obtenidos. Finalmente, en la Sección 5 se exponen las conclusiones y trabajos futuros.

2. MÉTODOS DE CLUSTERING

Los métodos expuestos se encuentran dentro de lo que se ha clasificado como hard clustering, y ofrecen soluciones sub-óptimas para el problema del clustering. En la primera parte se aborda el algoritmo K-Means, al cual se le da una especial atención debido a que su adaptación dentro del marco de la metaheurística Grasp será parte principal del presente trabajo. Luego, se hace una revisión de las metaheurísticas para el problema del clustering propuestas recientemente, dentro de las que podemos encontrar los Algoritmos Genéticos, los Algoritmo Meméticos y la Metaheurística Grasp.

2.1. Algoritmo K-Means

El algoritmo K-Means es una de las heurísticas comúnmente utilizadas para resolver el problema de clustering [18, 10]. La idea básica del algoritmo es obtener los K centros iniciales y formar clusters asociando todos los objetos de $\boldsymbol{X}$ a los centros más cercanos, después se recalculan los centros. Si esos centros no difieren de los centros anteriores, entonces el algoritmo termina; caso contrario, se repite el proceso de asociación con los nuevos centros hasta que no haya variación en los centros, o se cumpla algún otro criterio de parada como poco número de reasignaciones de los objetos.

Los K diferentes centros iniciales $\{\bar{\boldsymbol{x}}_j\}_{j=1,...,K}$ se seleccionan aleatoriamente de $\boldsymbol{X}$. La asociación del objeto $\boldsymbol{x}_i \in \boldsymbol{X}$ con el centro más cercano $\bar{\boldsymbol{x}}_j$ del cluster $\boldsymbol{C}_j$ es dada si $d(\boldsymbol{x}_i, \bar{\boldsymbol{x}}_j) < d(\boldsymbol{x}_i, \bar{\boldsymbol{x}}_p)$ para todo $j, p = 1, \ldots, K$ y $j \neq p$. Los centros son recalculados usando la expresión (2). La idea del algoritmo K-Means se presenta en psudocódigo, como:

Algoritmo K-Means$(\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}, K)$
1. Seleccionar aleatoriamente de $\boldsymbol{X}$ centros iniciales $\{\bar{\boldsymbol{x}}_i\}_{i=1,...,K}$
2. Para cada $\boldsymbol{x}_i \in \boldsymbol{X}$
2.1. Asociar $\boldsymbol{x}_i$ con el centro mas cercano:
$\boldsymbol{C}_j = \boldsymbol{C}_j \bigcup \{\boldsymbol{x}_i\}$, si $d(\boldsymbol{x}_i, \bar{\boldsymbol{x}}_j) < d(\boldsymbol{x}_i, \bar{\boldsymbol{x}}_p),\ \forall j, p = 1, ..., K$ y $j \neq p$
3. Fin para
4. Calcular los centros $\bar{\boldsymbol{x}}_i^* = \frac{1}{|\boldsymbol{C}_i|} \sum_{j=1}^{|\boldsymbol{C}_i|} \boldsymbol{x}_j$, para $\boldsymbol{x}_j \in \boldsymbol{C}_i$
5. Si no hay mas reasignaciones: $\bar{\boldsymbol{x}}_i^* = \bar{\boldsymbol{x}}_i,\ \forall i$, parar.
Caso contrario, considerar $\bar{\boldsymbol{x}}_i^*$ como nuevo centro $\bar{\boldsymbol{x}}_i$, e ir al paso 2.

Los principales inconvenientes del algoritmo K-Means citados por Peña et al. [23] son: su sensibilidad a la inicialización (debido a esto, el resultado final depende del estado inicial); el K-Means es un algoritmo de búsqueda local (el algoritmo minimiza la función objetivo dada en (1), pero no garantiza una configuración óptima de los clusters); y, se debe tener conocimiento previo del valor de K.

Peña et al. [23] discuten cuatro métodos para la inicialización del algoritmo K-Means. El primer método de inicialización es completamente aleatorio; el segundo es el método de Forgy [10]; el tercero es el método de McQueen [18]; y finalmente, el método de Kauffman y Rousseeuw [15]. Los tres primeros métodos son, de alguna manera, aleatorios; sólo el algoritmo propuesto por Kauffman y Rouseeuw es un algoritmo heurístico que identifica los objetos más representativos que prometen tener en su alrededor gran cantidad de objetos.

Peña et al. concluyen que la inicialización completamente aleatoria y la propuesta de Kauffman y Rouseeuw ofrecen una mejor inicialización para el algoritmo K-Means que el resto de métodos, haciéndolo más robusto.

2.2. Métodos Metaheurísticos

Los Algoritmos Genéticos han sido propuestos para el hard clustering por Murthy y Chowdhury [20], Bandyopadhyay y Maulik[1], y Pacheco y Valencia [21]. En [20] se codifican las soluciones en cromosomas de longitud igual al número de elementos de $\boldsymbol{X}$, por lo que el método es limitado por el número de elementos de $\boldsymbol{X}$. Las otras propuestas codifican los cromosomas con los valores de los centros de los clusters. En estos casos, un cromosoma está compuesto por un vector $\boldsymbol{a} = (\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_K)$, donde $\bar{\boldsymbol{x}}_i$ es centro de $\boldsymbol{C}_i$; la implementación requiere de menos recursos, y el método es factible para cualquier número de elementos de $\boldsymbol{X}$. Es decir, las propuestas [1] y [21], son parecidas en el aspecto de codificación de cromosomas, pero varían en cuanto a los operadores de *cruzamiento* y *mutación*, y que el primero utiliza el algoritmo K-Means para refinar la solución en cada generación de la población.

Los Algoritmos Meméticos han sido propuestos para el hard clustering por Pacheco y Valencia [21] y Merz [19]. Al igual que los Algoritmos Genéticos, los Algoritmos Meméticos, utilizan poblaciones de soluciones denominadas memes, que se van recombinando generación tras generación en búsqueda de un óptimo. La diferencia radica en que cada meme es obtenido por un algoritmo de búsqueda local en un espacio de soluciones de óptimos locales. En [19] se propone el uso del algoritmo K-Means para la generación de los óptimo locales, mientras que en [21] se realizan experimentos con diversos algoritmos de búsqueda local, tales como HK-Means y J-Means [12]. Tanto en [19], como en [21] se codifican los memes con los centros de los clusters; es decir, un meme es compuesto por un vector $\boldsymbol{a} = (\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_K)$ de centros obtenidos con un algoritmo de búsqueda local.

La metaheurística Grasp también ha sido propuesta para el problema del clustering por Cano et al. [6]. La metaheurística Grasp es un proceso de multiarranque compuesta por dos fases: fase de *construcción* en la que se generan buenas soluciones de manera aleatoria en base a un algoritmo goloso; y fase de *búsqueda local*, en que se busca una mejoría de las soluciones obtenidas en la fase de construcción. Cano et al. utilizan en la fase de construcción una adaptación del algoritmo de Kauffman y Rouseeuw [15] para encontrar los centros iniciales; y, en la fase de búsqueda local, usa el algoritmo K-Means para mejorar la solución encontrada en la fase previa. El método demuestra ser superior al algoritmo K-Means en las colecciones de datos usadas.

3. METAHEURÍSTICA GRASP PARA EL PROBLEMA DEL CLUSTERING

En parte, algunas inconveniencias del algoritmo K-Means fueron superadas con la ayuda de novedosas técnicas dentro de la optimización combinatoria, tales como Algoritmos Genéticos, Algoritmos Meméticos y Grasp. Pero aún continúan las inconveniencias, incluyéndose robustez de la convergencia a la solución e inicialización.

En esa perspectiva, proponemos una adaptación del algoritmo K-Means para la obtención de los centros iniciales, los cuales son alterados sino cumplen un criterio de evaluación establecido, y son refinados mediante un proceso de búsqueda local. Los procesos están diseñados dentro del marco de la metaheurística Grasp. En este sentido, dedicamos la siguiente subsección para presentar la arquitectura básica del Grasp (mayor detalle consultar en [9]), para un mejor entendimiento del uso de Grasp en Clustering.

3.1. Metaheurística Grasp

Un procedimiento de búsqueda voraz aleatoria y adaptativa (Grasp) es una metaheurística propuesta por Feo y Resende [9] para encontrar soluciones aproximadas de problemas de optimización combinatoria, mediante un proceso iterativo. En cada iteración se realizan dos fases de operaciones: *construcción* y *búsqueda local*. En la fase construcción se genera un conjunto solución $\boldsymbol{S}$ de una instancia $\boldsymbol{E}$ de un problema combinatorio, y en la fase de búsqueda local se determina una posible mejor solución a $\boldsymbol{S}$; finalmente, se elige la solución mejor entre la solución de la iteración anterior y la actual. La mejor solución será indicada por una función objetivo f. Cada iteración es realizada un número máximo de veces (MAX_ITER). A seguir se presenta en notación de pseudo-código o algoritmo Grasp básico tal como fue descrito.

Algoritmo Grasp (E, MAX_ITER, α)
1. Inicializar solución $S := \emptyset$ y $f^* := \infty$
2. Repetir MAX_ITER veces
2.1. Obtener una solución S^* de $Construccion\_Grasp(E, \alpha)$
2.2. Obtener una solución S^* de $Busqueda\_Local\_Grasp(S^*)$
2.3. Si $f(S^*) < f^*$, entonces
2.3.1. Actualizar $S := S^*$ y $f^* := f(S^*)$
2.4. Fin Si
3. Fin Repetir
4. Solución S

El hecho de que la fase de búsqueda local toma como entrada la solución obtenida en la fase de construcción proporciona un conocimiento frente a los algoritmos de búsqueda local tradicionales.

Construcción Grasp

En esta fase se construye un conjunto de soluciones en base a la adaptación de un algoritmo goloso. Tales algoritmos tienen una función de evaluación golosa $g\colon C \to R$ que selecciona el mejor elemento de un conjunto de candidatos C ha ser incorporados en la solución. El criterio de selección goloso de g depende del carácter del problema, puede ser maximización o minimización. El constructor de soluciones evita el determinismo de los algoritmos golosos, utilizando un parámetro de relajación α para formar una lista restringida de candidatos (Restricted Candidate List - RCL) alrededor del mejor elemento a seleccionar. El elemento ha ser incorporado en la solución es elegido aleatoriamente del RCL. Esta forma de selección proporciona al Grasp un aspecto estocástico de selección con tendencia a los mejores elementos que permite evitar los óptimos locales. El parámetro de relajación $\alpha \in [0, 1]$ indica la amplitud del RCL alrededor del mejor candidato. Cuando $\alpha = 0$, el RCL estará conformado sólo por el mejor candidato y la fase de construcción se comporta como un algoritmo goloso. Cuando $\alpha = 1$, el RCL estará conformado por el total de elementos de C y la selección de los candidatos será totalmente aleatoria. El mejor valor de α para el problema en estudio se obtiene a través de múltiples experimentos computacionales de calibración.

Búsqueda Local Grasp

La búsqueda local se realiza de manera iterativa, explorando en la vecindad de un conjunto de solución S generada por la operación de construcción. El desempeño de la operación de búsqueda local dependerá del método elegido. Si N es una vecindad de soluciones, se dice que $S' \in N(S)$ es un óptimo local si $f(S') < f(S)$. No existe un esquema de búsqueda local específico a utilizarse, solo es necesario que mejore la solución encontrada en la fase de construcción.

3.2. Grasp basado en K-Means

Adaptamos la metaheurística Grasp para resolver, de manera eficiente el problema del clustering minimizando la función objetivo dado en (1). En ese sentido, enfocamos el algoritmo K-Means considerando las dos fases de la arquitectura de Grasp y una fase adicional previa a estas, llamada *inicializacion_KM*.

El algoritmo formulado debe realizar repetidas veces (MAX_ITER veces) la secuencia de las tres fases mencionadas. Así, en la fase de inicialización (InicializacionKM) se obtienen los K centros iniciales y se definen los clusters $\boldsymbol{C}_j$ en torno de sus centros iniciales. Estos clusters, sirven de base para la fase de construcción (ConstruccionKM) donde se refina la solución inicial, evitando caer en óptimos locales. La siguiente fase, búsqueda de la mejor solución (MejoriaKM), se basa en la exploración de nuevas soluciones alterando heurísticamente la estructura de los clusters obtenidos en la fase de construcción para mejorar la solución. Por último, retiene la mejor solución entre la anterior y la actual.

Presentamos la estructura del algoritmo **GraspKM**, para después presentar en detalle cada uno de las fases mencionadas. Se considera como datos de entrada el conjunto de objetos $\boldsymbol{X}$, un número K de clusters a generar, la relajación α, y el máximo número de iteraciones MAX_ITER. Debemos esperar como resultado el conjunto de clusters $\boldsymbol{C}$.

Algortimo GraspKM $(\boldsymbol{X}, K, \alpha, MAX\_ITER)$
1. $f^* := \infty$, $\boldsymbol{C} := \{\}$
2. Repetir MAX_ITER veces
2.1. $\boldsymbol{C}' :=$ InicializacionKM($\boldsymbol{X}$, K)
2.2. $\boldsymbol{C}' :=$ ConstruccionKM($\boldsymbol{X}$, K, $\boldsymbol{C}'$, α)
2.3. $\boldsymbol{C}' :=$ MejoriaKM($\boldsymbol{X}$, K, $\boldsymbol{C}'$)
2.4. Si $f(C') < f^*$, entonces
2.4.1. $\boldsymbol{C} := \boldsymbol{C}'$
2.4.2. $f^* := f(C')$
2.5. Fin Si
3. Fin Repetir
4. Solucion $\boldsymbol{C}$

3.2.1. Configuración inicial

De manera similar al algoritmo K-Means, en esta fase se seleccionan K centros aleatoriamente, luego forma los clusters iniciales asociando el objeto $\boldsymbol{x} \in \boldsymbol{X}$ al cluster $\boldsymbol{C}_j$ si el centro $\bar{\boldsymbol{x}}_j$ es el menos distante al objeto. Finalmente, se calcula el nuevo centro o la media del cluster $\boldsymbol{C}_j$, $(j = 1, \ldots, K)$ haciendo uso de la expresión (2). A seguir, presentamos el algoritmo **InicializacionKM**.

InicializacionKM $(\boldsymbol{X}, K)$
1. Seleccionar K centros iniciales $\{\bar{x}_i = Random(\boldsymbol{X})\}_{i=1,\ldots,K}$
2. Para cada $\boldsymbol{x} \in \boldsymbol{X}$,
2.1. Asignar $\boldsymbol{x}$ a $\boldsymbol{C}_j$, cuando $j = ArgMin\{d(\boldsymbol{x}, \bar{\boldsymbol{x}}_i)\}_{i=1,\ldots,K}$
3. Fin Para
4. Calcular los centros $\{\bar{\boldsymbol{x}}_j := Media(\boldsymbol{C}_j)\}_{j=1,\ldots,K}$
5. Resultado $\boldsymbol{C} = \{\boldsymbol{C}_j\}_{j=1,\ldots,K}$

3.2.2. Construcción de soluciones

Esta fase es una adaptación del algoritmo K-Means (Sección 2.1) con el objetivo de evitar la convergencia a óptimos locales. La adaptación se realiza principalmente sobre la función golosa que asigna los objetos (paso 2.1 del algoritmo K-Means), la cual establece que un objeto $\boldsymbol{x} \in \boldsymbol{X}$ se asigna al cluster $\boldsymbol{C}_j$, si $\bar{\boldsymbol{x}}_j$ es el centro menos distante al objeto $\boldsymbol{x}$. Al aplicar el parámetro de relajación α sobre la función golosa, se crea un conjunto RCL de posibles clusters a los cuales puede ser reasignado un objeto. El RCL estará conformado por una vecindad alrededor del cluster más próximo al objeto evaluado. Esta fase se implementa mediante el algoritmo **ConstruccionKM** que se presenta a seguir, considerando como datos de entrada el número de clusters K, el conjunto de clusters $\boldsymbol{C} = \{\boldsymbol{C}_j\}_{j=1,\ldots,K}$ con sus respectivos centros $\{\bar{\boldsymbol{x}}_i\}_{i=1,\ldots,K}$ generados en la fase anterior, y el parámetro de relajación α.

ConstruccionKM $(\boldsymbol{X}, K, \boldsymbol{C}, \alpha)$
1. Repetir
1.1. Para cada $\boldsymbol{x} \in \boldsymbol{X}$ tal que $\boldsymbol{x} \in \boldsymbol{C}_j$ para algún $j = 1, \ldots, K$
1.1.1. $\overline{\beta} := Max\{d(\boldsymbol{x}, \bar{\boldsymbol{x}}_l) : d(\boldsymbol{x}, \bar{\boldsymbol{x}}_l) \leq d(\boldsymbol{x}, \bar{\boldsymbol{x}}_j)\}_{l=1,\ldots,K}$
1.1.2. $\underline{\beta} := Min\{d(\boldsymbol{x}, \bar{\boldsymbol{x}}_l)\}_{l=1,\ldots,K}$
1.1.3. $RCL := \{\boldsymbol{C}_t : d(\boldsymbol{x}, \bar{\boldsymbol{x}}_t) \leq \underline{\beta} + \alpha(\overline{\beta} - \underline{\beta})\}_{t=1,\ldots,K}$
1.1.4. $\boldsymbol{C}_t := Random(RCL)$
1.1.5. Si $t \neq j$
$\boldsymbol{C}_t := \boldsymbol{C}_t \cup \{\boldsymbol{x}\}$
$\boldsymbol{C}_j := \boldsymbol{C}_j - \{\boldsymbol{x}\}$
1.1.6. Fin de Si
1.2. Fin de Para
1.3. Recalcular centros $\{\bar{\boldsymbol{x}}_j := Media(\boldsymbol{C}_j)\}_{j=1,\ldots,K}$
2. Hasta que no haya mas reasignaciones
3. Resultado $\boldsymbol{C} = \{\boldsymbol{C}_j\}_{j=1,\ldots,K}$

En un proceso K-Means un objeto $\boldsymbol{x} \in \boldsymbol{C}_j$ será asignado a otro cluster $\boldsymbol{C}_p$ si la distancia $d(\boldsymbol{x}, \bar{\boldsymbol{x}}_p)$ es la mínima entre las distancias hacia los clusters y $j \neq p$. En el proceso ConstruccionKM, los posibles clusters que contendrían al objeto $\boldsymbol{x}$ en análisis son agrupados en un conjunto RCL que contiene un número menor de clusters cuyas distancias de sus centros al objeto $\boldsymbol{x}$ están en un intervalo definido por $\overline{\beta}$, que es el valor máximo de las distancias menores que la distancia a su centro de origen, $\underline{\beta}$ que es el mínimo de las distancias menores que la distancia a su centro de origen, regulada linealmente por el parámetro de relajación α. Del conjunto RCL será elegido aleatoriamente un cluster al cual será reasignado el objeto $\boldsymbol{x}$, desde luego retirándolo del cluster al cual correspondía antes de ese proceso. En los pasos que corresponden al segmento 1.1.4. del algoritmo ConstruccionKM son realizados las operaciones descritas. Esa operación de reasignación será realizada iterativamente para cada objeto $\boldsymbol{x} \in \boldsymbol{X}$. Después de la reasignación de todos los objetos de $\boldsymbol{X}$ en los diferentes clusters, es lógico que el centro haya variado; que justifica que, nuevamente, se deban recalcular los centros de cada cluster a través de la media aritmética de los objetos de los respectivos clusters, $\{\bar{\boldsymbol{x}}_j := Media(\boldsymbol{C}_j)\}_{j=1,\ldots,K}$, donde la función $Media$ es definida por (2).

Propuesto de esta manera, la fase de construcción evita el determinismo del algoritmo K-Means, teniendo una gran probabilidad de encontrar una mejor solución, aunque también puede encontrar peores. Es por ese motivo que en el procedimiento Grasp, la fase construcción, debe procesarse repetidas veces para obtener una gran variedad de soluciones que pueden ser mejoradas en la fase de búsqueda local.

3.2.3. Búsqueda de la mejor solución

La fase de construcción de soluciones, genera soluciones que no son necesariamente óptimos [9], debido a que la búsqueda se realiza de manera aleatoria en un espacio de soluciones restringido por el RCL. En la fase de búsqueda de la mejor solución, denominada MejoriaKM, se debe alterar la estructura de la solución generada en la fase de construcción con el fin de obtener una mejor solución.

Fränti y Kivijärvi [11] proponen un método de búsqueda local aleatorio para obtener una solución al problema del clustering. El método esta basado en un proceso de búsqueda local que altera la estructura de la solución generando un centro aleatorio, seleccionado del conjunto de objetos $\boldsymbol{X}$, y elimina el cluster con menor error cuadrático; luego, se reasignan los objetos a los clusters que tengan el centro más cercano; y finalmente, se realiza un proceso de refinamiento de la solución mediante el algoritmo K-Means. El proceso es repetido un número determinado de veces, y el mejor resultado es devuelto como la solución del problema. El método favorece la eliminación de clusters con menor error cuadrático que probablemente sea producto de alcanzar un óptimo local y evita este inconveniente a través de la generación de un nuevo centro.

Basados en la propuesta de Fränti y Kivijärvi diseñamos la fase MejoriaKM. La idea básica es, dado una solución, ignorar y regenerar clusters según ciertos criterios establecidos. A diferencia de Fränti y Kivijärvi, nuestra propuesta elimina el cluster que contiene menos objetos y genera un nuevo centro aleatoriamente dentro del cluster más disperso. Luego, todos los objetos de $\boldsymbol{X}$ son reasignados a los clusters más cercanos, de esta manera, los objetos del cluster eliminado son repartidos entre el resto de clusters y, a la vez, un cluster se forma alrededor del nuevo centro. La configuración de los cluster obtenida hasta este punto no es la óptima, por lo cual, la solución entra en un proceso de refinamiento, que para el caso será el mismo de la fase de construcción Grasp. Todo el proceso es repetido iterativamente hasta que no se pueda encontrar otra mejor solución. La forma heurística con que se modifica la estructura de la solución, obedece a la posibilidad de encontrar una mejoría, asumiendo que los clusters con menor cantidad de elementos y mayor dispersión ocurren porque el algoritmo alcanzó un óptimo local y que se puede encontrar una mejor solución.

En la mejoría descrita, podemos notar dos procesos que se pueden realizar de manera independiente. El primero de ellos es la alteración de la solución; es decir, la eliminación de un cluster, la generación de un nuevo cluster a partir de un centro elegido de manera aleatoria y la agrupación de los objetos alrededor del nuevo centro y de los existentes. A este proceso se le denominará en adelante de reagrupación. El segundo, es el proceso de refinamiento que corresponde al mismo ConstruccionKM lo que da un valor agregado a esta fase en cuanto a evitar alcanzar óptimos locales.

La estructura general del algoritmo MejoriaKM, se presenta a continuación. Considera como datos de entrada conjunto de datos $\boldsymbol{X}$, el número de clusters K y los clusters generados por la fase construcción $\boldsymbol{C} = \{\boldsymbol{C}_j\}_{i=1,\ldots,K}$, con respectivos centros $\bar{\boldsymbol{x}} = \{\bar{\boldsymbol{x}}_i\}_{i=1,\ldots,K}$. El proceso entra en una iteración hasta que se alcance una solución estable, realizando las dos etapas descritas anteriormente: la de reagrupación de los elementos de clusters críticos denominada ReagrupacionKM y el proceso ConstruccionKM usado para el refinamiento de la solución.

MejoriaKM $(\boldsymbol{X}, K, \boldsymbol{C}')$
1. Repetir
1.1. $\boldsymbol{C} := \boldsymbol{C}'$
1.2. $\boldsymbol{C}' :=$ ReagrupacionKM$(\boldsymbol{X}, K, \boldsymbol{C})$
1.3. $\boldsymbol{C}' :=$ ConstruccionKM$(\boldsymbol{X}, K, \boldsymbol{C}', \alpha)$
2. Mientras $(f(\boldsymbol{C}') < f(\boldsymbol{C}))$
3. Resultado $\boldsymbol{C} = \{\boldsymbol{C}_j\}_{j=1,\ldots,K}$

Reagrupación

El proceso de ReagrupacionKM requiere la identificación de los clusters $\boldsymbol{C}_l$ de menor número de elementos y $\boldsymbol{C}_h$ de mayor dispersión, y un nuevo centro $\bar{\boldsymbol{x}}_r$ elegido de manera aleatoria de los elementos pertenecientes a $\boldsymbol{C}_h$. Si $\boldsymbol{C}_l$ es el cluster con menor número de elementos, entonces l esta dado por

$$l = ArgMin\{|\boldsymbol{C}_j|\}_{j=1,\ldots,K}. \tag{3}$$

Para la identificación del cluster de mayor dispersión, se necesita primero el cálculo del error promedio del cluster, el cual esta dado por

$$E_j = \frac{\sum_{\boldsymbol{x}\in\boldsymbol{C}_j} d(\boldsymbol{x}, \bar{\boldsymbol{x}}_j)}{|\boldsymbol{C}_j|}. \tag{4}$$

Antes de identificar el cluster más disperso se describirán dos casos extremos, el primero de un cluster compacto y el segundo de un cluster disperso. El primer caso es un cluster con error promedio bajo y que tiene una buena cantidad de objetos, esta situación nos da la idea que el cluster esta bastante compacto. Por el contrario, si tenemos un cluster con error promedio alto y con gran cantidad de elementos entonces diremos que el cluster esta disperso. Es decir cuanto sea mayor la relación $\frac{E_j}{|\boldsymbol{C}_j|}$, mayor será la dispersión del cluster, por el contrario, menor será la dispersión y por consiguiente mayor será su compactación. La Figura 1 muestra la idea de un cluster compacto y otro disperso. Los clusters dispersos son los que nos interesa reagrupar, por ello se generará un nuevo centro dentro del cluster más disperso con la finalidad que contribuya a una mejor distribución de los objetos entre los clusters.

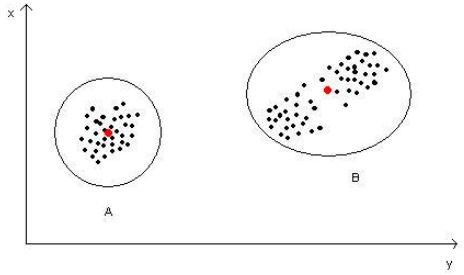


Figura 1: (A) cluster compacto, (B) cluster disperso

Entonces, si $\boldsymbol{C}_h$ es el cluster con mayor dispersión, h esta determinado por

$$h = ArgMax\left\{\frac{E_j}{|\boldsymbol{C}_j|}\right\}_{j=1,\ldots,K,j\neq l}. \tag{5}$$

El valor del nuevo centro $\bar{\boldsymbol{x}}_r$ es tomado desde $\boldsymbol{C}_h$, cuya selección se realiza de manera aleatoria y esta dado por $\bar{\boldsymbol{x}}_r = Random(\boldsymbol{C}_h)$.

Los cálculos descritos anteriormente se llevan a cabo dentro del algoritmo ReagrupacionKM, cuyo pseudocódigo se presenta a continuación, donde los datos de entrada son los mismos de MejoriaKM.

ReagrupacionKM $(\boldsymbol{X}, K, \boldsymbol{C})$
1. $l := ArgMin\{|\boldsymbol{C}_j|\}_{j=1,...,K}$
2. $h := ArgMax\left\{\frac{E_j}{|\boldsymbol{C}_j|}\right\}_{j=1,...,K,j\neq l}$
3. $\bar{\boldsymbol{x}}_r := Random(\boldsymbol{C}_h)$
4. Reemplazar $\bar{\boldsymbol{x}}_l$ por $\bar{\boldsymbol{x}}_r$
5. Para cada $\boldsymbol{x} \in \boldsymbol{X}$ // Generando clusters
5.1. asignar $\boldsymbol{x}$ a $\boldsymbol{C}_j$, donde $j = ArgMin\{d(\boldsymbol{x}, \bar{\boldsymbol{x}}_i)\}_{i=1,...,K}$
6. Fin Para
7. Calcular los centros $\{\bar{\boldsymbol{x}}_j := Media(\boldsymbol{C}_j)\}_{j=1,...,K}$
8. Resultado $\boldsymbol{C} = \{\boldsymbol{C}_j\}_{j=1,...,K}$

4. EXPERIMENTOS COMPUTACIONALES

Para validar nuestro método, realizamos una serie de experimentos computacionales con datos usados en [3, 14, 22, 15] y comparamos los resultados con otros métodos de la literatura.

4.1. Colecciones de Datos

Los datos que usamos para la evaluación de nuestro método son: Iris y Glass [3]; Crude Oil [14], Vowel [22] y Ruspini [15]. A continuación presentamos una breve descripción de las colecciones de datos usadas:

Iris. Consiste de 150 muestras plantas de lirio. El número de atributos es 4. El valor de K para este conjunto de datos es 3.

Glass. Consiste de 214 instancias de diferentes tipos de vidrio. El número de atributos es 9. El valor de K para este conjunto de datos es 7.

Crude Oil. Consiste de 56 muestras de petróleo crudo. Los datos presentan 6 atributos correspondientes al tipo de roca. El valor de K para este conjunto de datos es 3.

Vowel. Consiste de 871 muestras de seis clases de vocales del dialecto hindú Telugu. Los atributos corresponden a la medición de tres frecuencias. El valor de K para este conjunto de datos es 6.

Ruspini. Consiste de 75 vectores en dos dimensiones formando 4 grupos de manera natural. El valor de K para este conjunto de datos es 4.

4.2. Calibración del parámetro α

Los experimentos fueron realizados para cada uno de los conjuntos de datos considerando 4 distintos valores para α: 0.25, 0.50, 0.75 y 1.00. Con cada valor de α se realizaron 50 ejecuciones del proceso GraspKM con un valor de MAX_ITER de 100. Se debe tener en cuenta que, por las restricciones impuestas en la fase ConstruccionKM para la conformación del conjunto RCL, cuando $\alpha = 1$ no significará que la solución sea contruída de manera aleatoria, sino que el RCL estará conformado por todos los clusters cuyos centros sean mas cercanos a un objeto $\boldsymbol{x}$ que el centro de su cluster.

En en Cuadro 1 se resumem los resultados obtenidos para los conjuntos de datos descritos. El cuadro muestra el mejor valor, el peor valor y el promedio obtenido para la función objetivo f, en 50 ejecuciones de GraspKM. Nótese, que el mejor valor es alcanzado cuando $\alpha = 1$, por lo tanto es el valor que usará en las comparativas con otras metaheurísticas.

4.3. Comparación de resultados

Las comparativas serán con respecto al mejor valor encontrado para la minimización de la función objetivo (1) frente a otras dos metaheurísticas de la literatura de clustering: el Algoritmo Genético KGA-Clustering propuesto por Bandyopadhyay y Maulik [1] y el Algoritmo Grasp Grasp-KMeans propuesto por Cano et

	α	0.25	0.50	0.75	1.00
Iris	*Mejor Óptimo*	97.32594	97.32594	97.32594	**97.22213**
	Peor Óptimo	97.32594	97.32594	97.32594	97.257195
	Promedio	97.325966	97.325966	97.325966	97.226074
Glass	*Mejor Óptimo*	201.12637	201.12637	201.12637	**200.58238**
	Peor Óptimo	201.83528	201.83528	201.85007	201.85007
	Promedio	201.15884	201.17648	201.16087	201.02412
Crude Oil	*Mejor Óptimo*	279.27097	279.27097	279.27097	**278.96515**
	Peor Óptimo	279.27097	279.27097	279.27097	278.96515
	Promedio	279.27097	279.27097	279.27097	278.96515
Vowel	*Mejor Óptimo*	149374.36	149374.36	149374.36	**149350.52**
	Peor Óptimo	149388.86	149398.67	149388.86	149400.92
	Promedio	149381.16	149380.64	149381.42	149379.84
Ruspini	*Mejor Óptimo*	504.77338	504.77338	504.77338	**501.27274**
	Peor Óptimo	504.77338	504.77338	504.77338	501.27274
	Promedio	504.77338	504.77338	504.77338	501.27274

Cuadro 1: Calibración de α para la colección de datos Iris, Glass, Crude y Ruspini.

al [6]. Sabemos que cada método presenta sus características propias, el hecho de establecer el número de ejecuciones y/o el máximo de iteraciones similares es con la intención de establecer condiciones parecidas para la ejecución de los métodos, en todo caso, la comparación se realizará principalmente en los valores obtenidos para la función objetivo.

Bandyopadhyay y Maulik presentan los resultados obtenidos para 50 ejecuciones de KGA-Clustering con un máximo de 1,000 generaciones y población de 50 individuos para las colecciones de datos: Iris, Vowel y Crude Oil. Para las comparativas con el método GraspKM también consideraremos 50 ejecuciones con valor de $\alpha = 1$ y MAX_ITER de 1,000. Los resultados son resumidos en el Cuadro 2.

		K-Means	KGA-Clustering	GraspKM
Iris	*Mejor Óptimo*	97.32594	97.100777	97.22213
	Peor Óptimo	128.40422	97.100777	97.22213
	Promedio	104.26579	97.100777	97.22213
Crude Oil	*Mejor Óptimo*	279.270997	278.965150	278.965150
	Peor Óptimo	359.761992	278.965150	278.965150
	Promedio	284.248060	278.965150	278.965150
Vowel	*Mejor Óptimo*	149399.49	149356.01	149342.64
	Peor Óptimo	168764.32	149378.03	149374.36
	Promedio	154150.77	149368.45	149360.25

Cuadro 2: Comparativa entre KGA-Clustering y GraspKM para la colección de datos Iris.

En la tabla también se presenta los resultados para 50 ejecuciones del algoritmo K-Means, siendo superado por GraspKM en las tres colecciones de datos en cuanto al valor de la función objetivo y a la obtención de resultados más estables (véase promedio para K-Means y GraspKM). En comparación con KGA-Clustering, el método GraspKM se aproxima a KGA-Clustering en la colección de datos Iris. Precisamos decir, que para la colección Crude Oil tanto KGA-Clustering como GraspKM han obtenido los mismos resultados, y para la colección Vowel, GraspKM superó ampliamente los resultados de KGA-Clustering. Nótese que inclusive con un valor de $MAX\_ITER = 100$ (véase el Cuadro 1) se han tenido resultados comparables con el método KGA-Clustering.

Cano et al, presenta los resultados obtenidos para 10 ejecuciones del método Grasp-KMeans (con una máximo de 16 iteraciones) para las colecciones Glass y Ruspini. Para las comparativas con el método GraspKM también consideraremos 10 ejecuciones con $MAX\_ITER = 16$. Los resultados son resumidos en el Cuadro 3.

		K-Means	Grasp-KMeans	GraspKM
Glass	*Mejor Óptimo*	203.865	204.992	201.126
	Promedio	212.591	205.239	202.233
Ruspini	*Mejor Óptimo*	864.223	720.142	501.272
	Promedio	1178.064	720.142	502.028

Cuadro 3: Comparativa entre Grasp-KMeans y GraspKM para la colección de datos Glass y Ruspini.

El valor del *Peor Óptimo* no es mostrado por Cano et al, por tanto no hay punto de comparación sobre este valor. En tanto que, para los valores de *Mejor Óptimo* y *Promedio* el método GraspKM demuestra ser superior a Grasp-KMeans en las colecciones Glass y Ruspini.

5. CONCLUSIONES Y TRABAJOS FUTUROS

El presente artículo propone una metaheurística GRASP denominada GraspKM para el problema del clustering. El método está basado en los principios del algoritmo K-Means y ha demostrado ser superior al algoritmo K-Means y otras metaheurísticas. El método GraspKM se concentra en dos de los principales inconvenientes del algoritmo K-Means. El incoveniente de la convergencia a óptimos locales es acometida con la forma golosa adaptativa de construir soluciones. Esta característica a su vez, alivia el inconveniente de la inicializacion, al generar soluciones iniciales de buena calidad, que ayuda a la fase MejoríaKM a encontrar una mejor solución.

En la fase MejoriaKM se introduce el concepto de dispersión de un cluster, de manera que la reasignación de sus objetos bajo un criterio heurístico favorece a mejorar el valor de la función objetivo f.

La meteheurística Grasp es una secuencia repetitiva de procesos, de manera que, a mayor cantidad de ejecuciones existe la posibilidad de encontrar mejores soluciones. El clustering por lo general debe manejar grandes cantidades de objetos. En ese sentido, el uso de estructuras de datos que agilicen los accesos a los objetos (por ejemplo KD-Trees) pueden ser usados. También una reducción del número de cálculos puede ser aplicada para mejorar los tiempos de respuesta cuando la cantidad de objetos es grande (derivados del teorema de la desigualdad de triángulos).

Agradecimientos

A la facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional Mayor de San Marcos de Lima - Perú, por el apoyo brindado para la publicación de esta investigación.

Referencias

[1] Bandyopadhyay, S., Maulik, U. An evolutionary technique based on K-Means algorithm for optimal clustering in R^n. *Information Sciences*. Vol. 146 (1-4), 2002, pp. 221–237.

[2] Bathia, S., Deogun, J. Conceptual Clustering in Information Retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. Vol 28 (3), 1998, pp. 427–436.

[3] Blake, C., Merz, C. UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[4] Bezdek, J. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York, 1981.

[5] Brucker, P. On the Complexity of Clustering Problems. *Lecture Notes in Economics and Mathematical Systems*. Vol. 157, 1978, pp. 45–54.

[6] Cano J., Cordón, O., Herrera, F., Sánchez L. Greedy Randomized Adaptive Search Procedure Applied to the Clustering Problem as an Initialization Process Using K-Means as a Local Search Procedure. *International Journal of Intelligent and Fuzzy Systems*. Vol. 12, 2002, pp. 235–242.

[7] Dempster, A., Laird, N., Rubin, D. Maximum-likelihood from imcomplete data via the em algorithm. *Journal of the Royal Statistical Society B*. Vol. 39, 1977, pp. 1–39.

[8] Fayyad, U., Piatetsky-Shapiro, G., Padhraic S. From Data Mining to Knowledge Discovery in Databases. *American Association for Artificial Inteligence*. 1996, pp. 37–54.

[9] Feo, T., Resende, M. Greedy Randomized Adaptative Search Procedure. *Journal of Global Optimization*. Vol. 6, 1995, pp 109–133.

[10] Forgy, E. Cluster analysis of multivariate data: Efficiency vs. Interpretability of classifications. *Biometrics*. Vol. 21 (768), 1965.

[11] Fränti, P., Kivijärvi, J. Randomised Local Search Algorithm for the Clustering Problem. Springer-Verlag London Limited. *Pattern Analysis and Applications*. Vol 3, 2000, pp. 358–369.

[12] Hansen P., Mladenovic N. J-Means: A new local search heuristic for minimun sum-of-squares clustering. *Pattern Recognition*. Vol. 34 (2), 2001, pp. 405–413.

[13] Jain, A., Murty, M. Flynn P. Data Clustering a Review. *ACM Computer Surveys*. Vol. 31 (3) 1999, pp. 264–323.

[14] Johnson, R., Wichern, D. Applied Multivariate Statistical Analysis. Prentice Hall, Englewood Cliffs, NJ. 1982.

[15] Kaufman, L., Rousseeuw, P. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley and Sons. New York, NY. 1965.

[16] Kearns, M., Mansour, Y., Ng, A. An Information-Theoretic Analysis of Hard and Soft Assignment Methods for Clustering. *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann. 1997, pp. 282–293.

[17] Makhoul, J., Roucos, S., Gish, H. Vector quantization in speech coding. *Preccedings of the IEEE*. Vol. 73, 1985, pp. 1551–1558.

[18] McQueen, J. Some methods for classification and analysis of multivariate observations. *In Preccedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. 1967, 281–297.

[19] Merz, P. Analysis of gene expression profiles: an application of memetic algorithms to the minimum sum-of-squares clustering problem. *BioSystems*. Vol. 72 (1-2), 2003, pp. 99–109.

[20] Murthy, C., Chowdhury, N. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*. Vol. 17, 1996, pp. 825–832.

[21] Pacheco, J., Valencia, O. Design of hybrids for the minimim sum-of-squares clustering problem. *Computational Statistics & Data Analysis*. Vol. 43 (2), 2003, pp. 235–248.

[22] Pal, S., Dutta, D. Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 7, 1977, pp. 625–629.

[23] Peña, J., Lozano J., Larrañaga, P. An empirical comparison of four initialization methods for the K-Means algorithm. *Pattern Recognitioin Letters*. Vol. 20, 1999, pp. 1027–1040.

[24] Pham, D., Prince, J. An Adaptive Fuzzy C-Means Algorithm for Image Segmentation in the Presence of Intensity Inhomogeneities. *Pattern Recognition Letters*. Vol. 20 (1), 1999, pp. 57–68.

[25] Scheunders, P. A genetic Lloyd-Max image quantization algorithm. *Pattern Recognition Letters*. Vol. 17 (5), 1996, pp. 547–556.

[26] Solberg, A., Taxt, T., Jain, A. A markov random field model for classification of multisource satellite imagery. *IEEE Trans. Geoscience and Remote Sensing*. Vol. 34 (1), 1996, pp. 100–113.

A Lexical and Syntactical Analyzer for the Exporting of QNAP*to the Performance Model Interchange Format (PMIF).

Jerònia Rosselló, Catalina M. Lladó, Ramon Puigjaner
Universitat de les Illes Balears
Departament de Matemàtiques i Informàtica
Cra de Valldemossa, Km. 7.6, 07071
07071, Palma de Mallorca, Spain.
jeronia.rossello@vodafone.es, cllado@uib.es, putxi@uib.es.

Connie U. Smith
Performance Engineering Services
PO Box 2640 Santa Fe
New Mexico, 87504-2640 USA
www.perfeng.com

Abstract

A Performance Model Interchange Format (PMIF) provides a mechanism whereby system model information may be transferred among performance modeling tools. The PMIF allows diverse tools to exchange information and requires only that the importing and exporting tools either support the PMIF or provide an interface that reads/writes model specifications from/to a file. This paper presents the development of the Qnap exporting mechanism to the PMIF. Since access to Qnap internal code is not possible, the only way of exporting Qnap is by translating Qnap input files, for which a lexical and syntax analyzer needs to be developed.

Keywords: Software Performance Engineering, Tool Interoperability, Performance Model, XML, Queueing Network Model, Interchange Format, Compiler Techniques.

Resumen

Un Formato de Intercambio de Modelos de Rendimiento (PMIF, *Performance Model Interchange Format*) proporciona un mecanismo para el intercambio de información de modelos de rendimiento entre diferentes herramientas de modelado. El PMIF permite que varias herramientas intercambien información solo requiriendo que las herramientas que exportan e importan modelos soporten el PMIF o dispongan de una interfaz que lee/escribe las especificaciones en PMIF desde/a un fichero. Este artículo presenta el desarrollo del mecanismo de exportación de Qnap a PMIF. Dado que el acceso al código interno de la herramienta Qnap no resulta viable, la única manera de exportar Qnap es traduciendo los ficheros de entrada de Qnap, por lo cual se necesita desarrollar un analizador léxico y sintáctico.

Palabras claves: Ingeniería de Rendimiento de Software, Interoperabilidad de Herramientas, Modelos de Rendimiento, XML, Redes de Colas, Formatos de Intercambio, Técnicas de Compilación.

1 INTRODUCTION

(PMIF) [10, 7] is a common representation for Queuing Network Model (QNM) data that can be used to move models among modeling tools. A user of several tools that support the format can create a model in one tool, and later move the model to other tools for further work without the need to laboriously translate from

*Qnap is a commercial tool developed by Simulog [5] for queueing networks modelling.

one tool's model representation to the other, and the need to validate the resulting specification. Tools that support the PMIF format only need to implement the export and import mechanisms to the PMIF rather than implement customized exports and imports for every other tool that they want to share information with[1].

PMIF users could, for example, compare solutions from multiple tools; create input specifications in PMIF or in a familiar tool rather than learn the interface to multiple tools; migrate a model to temporarily use another tool to study more detailed models; and create software performance models to study architecture and design trade-offs, then use another tool to study details of the computer system.

Depending on the modeling tool and the availability of its source code, its exporting and importing mechanisms to/from PMIF can vary from a simple implementation to a huge engineering project. In [7] a prototype is described in which the exporting tool is *SPE·ED* and the importing tool is Qnap [2, 6] . In this case, both importing and exporting mechanisms were quite straightforward. On the contrary, when thinking about the exporting mechanism from Qnap to PMIF, one finds out that unfortunately it is not so simple.

Due to the fact that we do not have access to Qnap's internal code, the Qnap to PMIF export mechanism can only be achieved by using the Qnap input file that defines the QNM in Qnap's format. Such a file needs to be carefully analyzed in order to be translated. As a consequence, the translation consists of a much more laborious process (compared to for instance the use of the Document Object Model (DOM) in the export mechanism from *SPE·ED* [8]) that needs to be carefully studied and designed. This process includes the use of compiler techniques such as lexical and syntactical analysis (see the Appendix for an overview of these techniques). This paper presents the analysis, design and implementation of such a process.

The rest of the paper is organized as follows. Section 2 gives an overview of the PMIF format, describing its main components and how they relate. Following, Section 3 considers the most importat issues that had to be addressed in the design of the Qnap to PMIF exporting mechanism while Section 4 describes its implementation. Section 5 shows an example of the application of the translation process step by step. Finally, Section 6 reports some future work and conclusions. The Appendix provides an overview of compiler techniques and tools for lexical analysis and syntactic analysis.

2 PMIF OVERVIEW

PMIF was first defined using an EIA/CDIF (Electronic Industries Association/CASE Data Interchange Format) paradigm that calls for defining the information requirements for a Queueing Network Model (QNM) with a meta-model [9, 10], that is, it is a model of the information that goes into constructing a QNM. A transfer format was then created from the meta-model and used to exchange information. The PMIF allows diverse tools to exchange information and requires only that the importing and exporting tools either support the PMIF or provide an interface that reads/writes model specifications from/to a file.

A new version of the meta-model has been recently defined together with a new PMIF specification (PMIF 2.0) [7, 8], which is implemented as an XML (Extensible Markup Language) schema [12].

The diagram of the XML Schema for PMIF 2.0 is in Figure 1. The diagram shows that a *QueueingNetworkModel* is composed of one or more *Nodes*, and one or more *Workloads*. A *Server* provides service for one or more *Workloads*. A *Workload* represents a collection of transactions or jobs that make similar *ServiceRequests* from *Servers*. There are two types of *Workloads*: *OpenWorkload* and *ClosedWorkload*.

A *ServiceRequest* specifies the average *TimeService*, *DemandService* or *WorkUnitService* for each *Workload* that visits the *Server*. A *TimeServiceRequest* specifies the average service time and number of visits. A *DemandServiceRequest* specifies the average service demand (service time x number of visits). A *WorkUnitServiceRequest* specifies the average number of visits requested by each *Workload* that visits a *WorkUnitServer*. Upon completion of the *ServiceRequest*, the *Workload Transits* to other *Nodes* with a specified probability.

More detailed information and the PMIF Schema definition can be found in [7, 8].

In addition, a prototype was also developed, in which the exporting tool is *SPE·ED* and the importing tool is Qnap. *SPE·ED* uses the Document Object Model (DOM) [12] to export the pmif.xml. On the other hand, since the access to Qnap source code was not provided, an XSLT specification that transforms a pmif.xml file into a file that is read and executed by Qnap was implemented.

[1]It is possible to use the PMIF without an explicitly coded import and export function as long as the tool provides a file input/output capability.

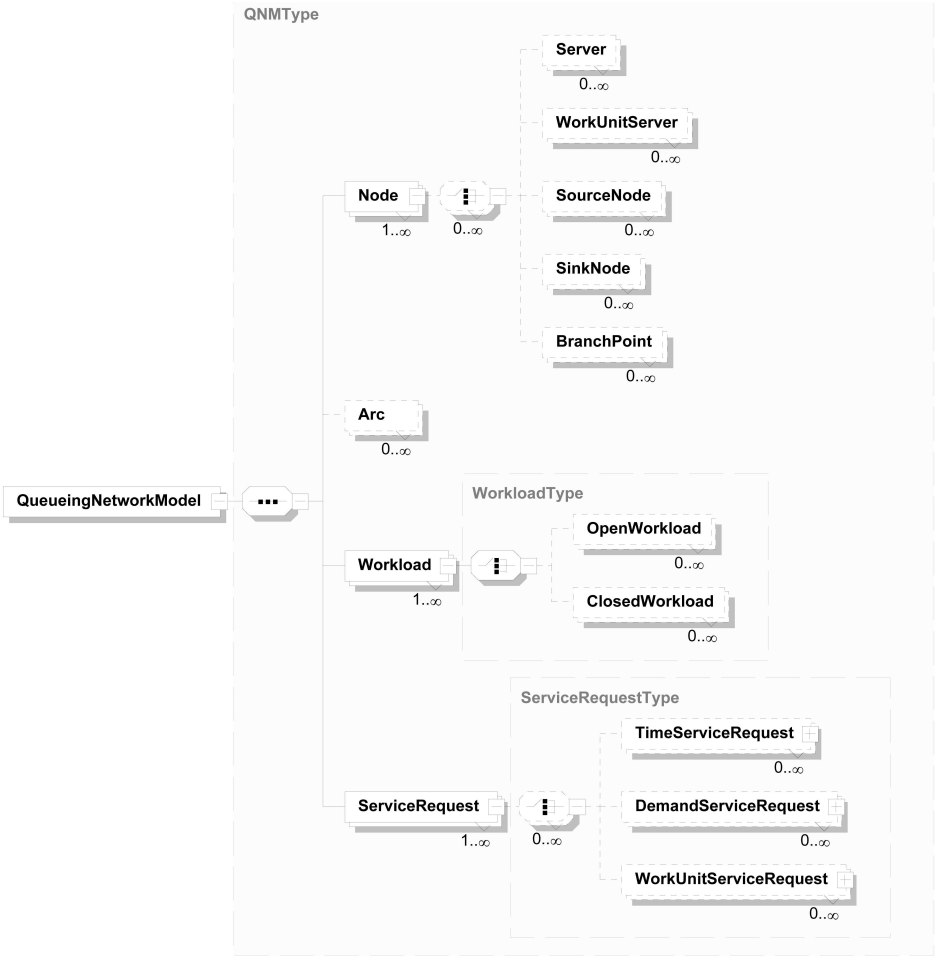


Figure 1: XML Schema for PMIF 2.0

Moreover, in [3] the design and implementation of a PMIF Web service for the modeling tool Qnap is presented as a further step of the transformation from PMIF to Qnap.

3 QNAP TO PMIF EXPORTATION MECHANISM

As previously stated, we need to develop a mechanism to export a Qnap model to the PMIF format. Due to the fact that we do not have access to Qnap internal code we can only do it through the design and implementation of a mechanism to transform Qnap input files into files in PMIF format. This transformation can be achieved by means of a lexical analyzer and a syntax analyzer.

A lexical analyzer needs to know the regular expressions that form the language and a syntax analyzer needs to know the grammar that the language uses (see the Appendix for more detailed information). In our case, the regular expressions and the grammar are a subset of those that form the Qnap language [6]. The description and justification of them is detailed in subsection 3.1.

The syntax and lexical analysis of the Qnap input file gives the information needed to generate the PMIF file. However, it is convenient to do the analysis of the Qnap file and the generation of the PMIF file in two separate phases. Subsection 3.2 justifies the need for two phases and explains their design and interaction.

3.1 Qnap Modified Regular Expressions and Grammar

The Qnap language has its own reserved words, types and grammar [6]. The reserved words and types can be detected by using regular expressions, so a Qnap compiler uses a specific set of regular expressions. However, Qnap is a modelling tool that allows the user to solve more cases and in more ways, than the PMIF covers. Therefore, the Qnap grammar and its regular expressions have been reduced so that only what is admissible by the PMIF specification is admitted.

The regular expressions admitted are:

- The ones with numeric format, i.e. with float or exponential notation, which follow:

  NUMBER $= [1-9][0-9]*$

  FLOAT $=[0-9]*\backslash\cdot[0-9]+$

  IDENT $= ([a-z])([0-9]|[a-z])*$

  EXPONEG $= [0-9]*\backslash\cdot[0-9]+\backslash E\backslash-[0-9]*$

  EXPOPOS $= [0-9]*\backslash\cdot[0-9]+\backslash E\backslash+[0-9]*$

  EXPONEGCURTA $= [0-9]*\backslash E\backslash-[0-9]*$

  EXPOPOSCURTA $= [0-9]*\backslash E\backslash+[0-9]*$

- Specific Tokens used in the Qnap language: /STATION/, /DECLARE/, /END/, NAME, =, TYPE, SERVICE, EXP, (,), ;, , , SERVER, SOURCE, SINGLE, MULTIPLE, INFINITE, INTEGER, REAL, QUEUE, CLASS, STEP, UNTIL, FIFO, PS, SCHED, INIT, TRANSIT, OUT.

The following are some restrictions that the modified grammar imposes:

- Everything that in Qnap is specific to simulation programs (for instance, the *customer* object type or the *flag* object) does not need to be included in our grammar since the PMIF currently does not include those.
- If a Qnap model has only one class of clients, the class declaration is optional. In the new grammar, at least one class needs to be defined. Moreover, all the classes that are going to be used in the model need to be declared at the beginning and before any node (station) is declared.
- Qnap also allows the change of class for a client that goes from one node to another which is not allowed in the PMIF or the new grammar.
- Qnap permits the definition of two nodes as initial nodes for the same class. This is not covered either in our modified grammar.

The resulting modified grammar can be seen as a sub-grammar of the Qnap grammar and it is shown below:

program ⟶ GeneralBlocList /END/
GeneralBlocList ⟶ BlocList
BlocList ⟶ ComandDeclare BlocListStation
BlocListStation ⟶ BlocListStation ComandStation | ComandStation
ComandDeclare ⟶ /DECLARE/ VariableDeclareList
VariableDeclareList ⟶ VariableDeclareList PartVariableDeclare
| PartVariableDeclare
PartVariableDeclare ⟶ CLASS IdentifierClassList ;
| QUEUE IdentifierQueueList ;
| INTEGER IdentifierValueList ;
| REAL IdentifierValueList ;
IdentifierQueueList ⟶ IdentifierQueueList , QueueSize
| QueueSize
QueueSize ⟶ identifier | identifier (number)
IdentifierClassList ⟶ IdentifierClassList , identifier
| identifier

IdentifierValueList ⟶ IdentifierValueList , identifier ValueAssignment
| identifier ValueAssignment
ValueAssignment ⟶ = number | = double | void
ComandStation ⟶ /STATION/ ParameterName ParametersStationList
ParameterName ⟶ NAME = llistaIdentificadorsStation , identifier
| IdentifiersStationList , identifier (number)
| IdentifiersStationList , identifier (number STEP number UNTIL number)
| identifier
| identifier (number)
| identifier (number STEP number UNTIL number)
ParametersStationList ⟶ ParametersStationList ParameterStation
| ParameterStation
ParameterStation ⟶ ParameterType
| ParameterService
| ParemeterSched
| ParameterInit
| ParameterTransit
ParameterType ⟶ TYPE = StationType ;
StationType ⟶ SERVER | SERVER , multiple | SOURCE | multiple
multiple ⟶ SINGLE | MULTIPLE (number) | INFINITE
ParameterService ⟶ SERVICE ClassList = Distribution ;
Distribution ⟶ EXP (number) | EXP (double)
ClassList ⟶ (IdentifierList) | void
IdentifierList ⟶ IdentifierList , identifier | identifier
ParameterSched ⟶ SCHED = SchedType ;
SchedType ⟶ FIFO | PS
ParameterInit ⟶ INIT ClassList = number ;
ParameterTransit ⟶ TRANSIT ClassList = routing ;
routing ⟶ queue | PairList final
PairList ⟶ PairList , pair | pair
pair ⟶ queue , number | queue , double
final ⟶ , queue | void
queue ⟶ identifier | identifier (number) | OUT

3.2 Modular Design

The lexical analyzer can carry out some actions depending on the tokens found (see the Appendix for a detailed explanation). These actions could create a DOM object and afterwards generate the PMIF output file from it. However, the structure of the QNAP language combined with the characteristics of the DOM would make the code included in the analyzer long and complicated (some of these are detailed below). This is the reason why we developed two modules that carry out the transformation in two phases. The first one applies the syntax and lexical analysis to the Qnap input file and generates memory structures containing the information necessary for a second phase which is in charge of generating the XML file. The process is shown in Figure 2, in which the syntax analyzer (SA) asks the lexical analyzer (LA) for a token (*tk?*) and the lexical analyzer responds with a token. $Obj1 \cdots ObjN$ are the memory structures or objects. When the first phase finishes, it has obtained the characteristics of the model and saved them into the memory structures. The second phase works on these structures to generate the PMIF XML output.

The main Qnap characteristics that make the generation of memory structures convenient while carrying out the analysis are:

- Qnap has default values for most of the model parameters. The memory structures are initialized with those default values.
- Qnap allows the update of those values at any point in the model specification and as many times as wanted (and it is very usual to find this sort of programming in Qnap programs). The last one found (the one that appears latest in the input file) is the value used. So, for instance, a node (STATION

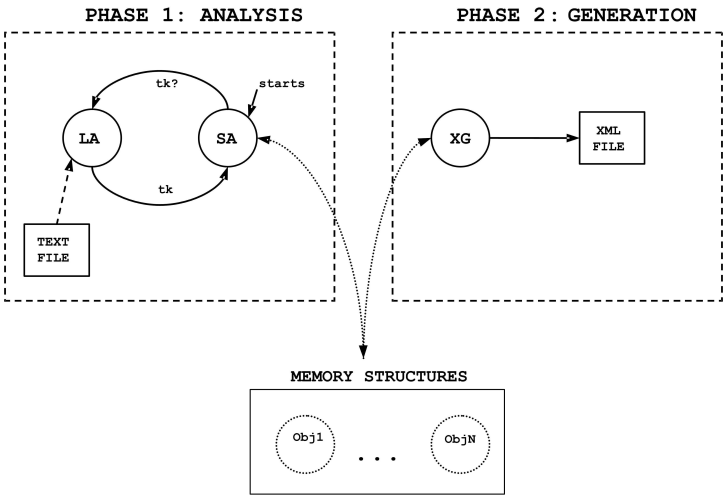


Figure 2: 2-Phases Design. LA: lexical analyzer, SA: syntax analyzer, XG: XML-Generator, tk: token

in Qnap) can be redefined as many times as wanted and the last definition (seeing the input file sequentially) is the valid one. Using the intermediate memory structures, these are directly modified every time a modification of the model definition is found.

Qnap models often have redefinitions of nodes and attributes in the input file, thus the overhead is higher for updating a DOM document tree than it is for the intermediate memory structures.

4 IMPLEMENTATION OF THE MECHANISM

The implementation of the two phases described in the previous section starts with the use of JLex [11] and JavaCup [4] tools (described in the Appendix) to create the lexical analyzer and syntax analyzer respectively. JLex uses the file with the Qnap regular expressions as the input file. JavaCup uses the grammar described in subsection 3.1 as input plus the required actions needed to create the memory structures that contain the information of the model.

The lexical analyzer generated by JLex carries out the subsequent actions in this specific order:

- Groups the characters of the Qnap input file into tokens and passes them to the syntax analyzer. These tokens are manipulated with the Symbol class, so the objects sent to the syntax analyzer are Symbols (java specification of tokens). Each token or Symbol is composed of the type of the token (or name) and the value of the token.
- Detection of the regular expressions subset corresponding to the ones with numeric format (detailed in section 3) and detection of identifiers. The value assigned to the numeric tokens (value of Symbol) is the value of the number, and the name of the token is *Double* or *Integer*. In the case of identifiers, the value of Symbol is the name of the string and the type is *identifier*.
- Detection of the rest of the tokens. In this case, the name (type) of the Symbol is similar to the name of the token detected and the value is the corresponding value found in the Qnap input file.

The syntax analyzer generated is able to check whether the input adheres to the grammar described in subsection 3.1. Its tasks can be briefly described as the following:

- Queries the lexical analyzer for tokens, and detects whether the sequence of received tokens is valid against the grammar.
- While the grammar productions are being reduced, the memory structures (described below) are filled with the corresponding values.

- Once all the grammar is reduced, the filling of the memory structures is finished and the phase two (XML Generation) can start.

Subsequently, the memory structures are described in detail. Three tables form these structures: the Nodes Table Information (NTI), the Routing Table Information (RTI) and the Workload Table Information (WTI).

The NTI saves the characteristics of the nodes in the system (STATIONs in Qnap). This table has as many rows as the system nodes and it has $3 + N$ columns, where N is the number of workloads (CLASSes in Qnap) defined in the model. The first column contains the type of the node, the second column the scheduling policy and the third column the quantity of servers. Finally, the following N columns contain the service time for each specific workload. This table is implemented as a 2D-vector as shown in Table 1.

	Type	Sched	Quantity	Workload1	Workload2	...	WorkloadM
Node1							
Node2							
...							
NodeN							

Table 1: Nodes Table Information

The RTIs save the routing probabilities (TRANSITs in Qnap) of all the nodes of the model. There is a RTI for each workload. Given a workload, the RTI values represent the routing probabilities between all the pair of nodes of the system. The set of RTIs is implemented as a 3D-vector. The 2-D part corresponding to 1 workload is shown in Table 2.

Workload1	Node1	Node2	...	NodeN
Node1				
Node2				
...				
NodeN				

Table 2: Routing Table Information

The WTI saves the workload characteristics for open and closed workloads. For closed workloads only the first part of the table will be filled. On the contrary, for open workloads only the second part of the table will be used. Table 3 shows these parts, implemented as a 2D-vector.

	ClosedWorkload			OpenWorkload		
	NumberOfJobs	ThinkDevice	ThinkTime	ArrivalRate	ArrivalsAt	DepartsAt
Workload1						
Workload2						
...						
WorkloadN						

Table 3: Workload Table Information

The second phase (see Fig. 2) or XML generation is in charge of generating a DOM structure by querying the three previously described tables, and posteriorly, creating the PMIF XML document. This functionality involves 3 well defined steps:

- Initialization of the DOM structure with the basis: A node called *QueueingNetworkModel*, including two mandatory child nodes: *Node* and *Workload* nodes [7, 8]. So that, it includes:
  $< QueueingNetworkModel >$
  $< Node/ >$
  $< Workload/ >$
  $< /QueueingNetworkModel >$

- Implementation of the four methods which are in charge of creating nodes, workloads, requests and arcs (described in Table 4). All of them get data from the table structures described above and add the nodes with the required attributes and children to the DOM structure.

- Generation of the PMIF XML file from the DOM structure.

NodesGeneration	Adds as many children to $< Node/ >$ as nodes are in the NTI each with its attributes. Children can be *SourceNode*, *SinkNode*, *WorkUnitServer* or *Server*
WorkloadsCreation	Adds as many children to $< Workload/ >$ as workloads are in the WTI. A child can be *ClosedWorkload* or *OpenWorkload*, each one of them with their own attributes
ArcsCreation	Creates as many *Arc* nodes as TRANSITs are in the RTI's
RequestCreation	Creates *ServiceRequest* which will contain *TimeServiceReq* nodes by querying the NTI and the RTI

Table 4: Description of the Methods used to generate the DOM structure

5 USE CASE

This section describes the Qnap to PMIF exporting mechanism step by step. The example used is based on the ATM model from [10]. The Qnap code (input file to be transformed) corresponding to this model is shown below:

```
/DECLARE/   QUEUE cpu;
            QUEUE atm, disks;
            QUEUE sourcen1, sourcen2 ;
            CLASS withdraw, getbala;
            REAL twithdra, tgetbal;

/STATION/   NAME= cpu;
            SCHED = PS;

/STATION/   NAME = atm;
            SERVICE = EXP(1);
            TYPE = INFINITE;

/STATION/   NAME = disks;
            SERVICE = EXP(0. 05);
            SCHED = FIFO;

/STATION/   NAME = sourcen1;
            TYPE= SOURCE;
            SERVICE= EXP(1);
            TRANSIT(withdraw)= cpu, 1;

/STATION/   NAME = sourcen2;
            TYPE= SOURCE;
            SERVICE= EXP(1);
            TRANSIT(getbala) = cpu, 1;

/STATION/   NAME = atm;
            TRANSIT(withdraw) = cpu, 1 ;

/STATION/   NAME = disks;
            TRANSIT(withdraw) = cpu, 1 ;

/STATION/   NAME = atm;
            TRANSIT(getbala) = cpu, 1 ;

/STATION/   NAME = disks;
            TRANSIT(getbala) = cpu, 1 ;

/STATION/   NAME = cpu;
            SERVICE(withdraw) = EXP(0.000315);
            TRANSIT(withdraw) = atm, 0.55, disks, 0.4, OUT , 0.05 ;

/STATION/   NAME = cpu;
            SERVICE(getbala) = EXP(0.00025);
            TRANSIT(getbala) = atm, 0.6, disks, 0.3, OUT , 0.1 ;
```

/CONTROL/
/END/

The lexical analyzer detects the tokens and the parser proves that the token sequence complies with the grammar and at the same time generates the tables (see section 4) and fills them with the values found. For this example the tables are filled as shown below:

	Type	Sched	Quantity	Withdraw	Getbala
Sourcen1	Source	PS	1	1.0	0
Sourcen2	Source	PS	1	0	1.0
Disks	Single	FCFS	1	0.05	0.05
Atm	Infinite	IS	1	1.0	1.0
Cpu	Single	PS	1	3.15E-4	2.5E-4
OUT	Single	PS	1		

Table 5: Nodes Table Information for the ATM example

Withdraw	Sourcen1	Sourcen2	Disks	Atm	Cpu	OUT
Sourcen1	0.0	0.0	0.0	0.0	1.0	0.0
Sourcen2	0.0	0.0	0.0	0.0	0.0	0.0
Disks	0.0	0.0	0.0	0.0	1.0	0.0
Atm	0.0	0.0	0.0	0.0	1.0	0.0
Cpu	0.0	0.0	0.4	0.55	0.0	0.05
OUT	0.0	0.0	0.0	0.0	0.0	0.0

Table 6: Routing Table Information for the ATM example and the Withdraw workload

Getbala	Sourcen1	Sourcen2	Disks	Atm	Cpu	OUT
Sourcen1	0.0	0.0	0.0	0.0	0.0	0.0
Sourcen2	0.0	0.0	0.0	0.0	1.0	0.0
Disks	0.0	0.0	0.0	0.0	1.0	0.0
Atm	0.0	0.0	0.0	0.0	1.0	0.0
Cpu	0.0	0.0	0.3	0.6	0.0	0.1
OUT	0.0	0.0	0.0	0.0	0.0	0.0

Table 7: Routing Table Information for the ATM example and the Getbala workload

Both Tables 6 and 7 show one row completely empty, *Source1* and *Source2* respectively. This is normal considering the Qnap restriction of only allowing 1 class (or workload) for source node.

When the parser has finished and the structures are filled the second phase starts with the generation of the DOM structure by querying the three previously shown tables. The XML file in PMIF format is afterwards created resulting with the following:

$<?xmlversion = "1.0"encoding = "UTF-8"?>$
$<QueueingNetworkResultsxmlns:xsi = "http://www.w3.org/2001/XMLSchema-isntance"$
$xsi:noNamespaceSchemaLocation = "http://www.perfeng.com/pmif/pmifschema.xsd"Name = "ATM">$
$<Node>$
$<SourceNode\ Name = "sourcen2"/>$
$<SourceNode\ Name = "sourcen1"/>$
$<Server\ Name = "disks"\ Quantity = "1"\ SchedulingPolicy = "FCFS"/>$
$<Server\ Name = "atm"\ Quantity = "1.0"\ SchedulingPolicy = "IS"/>$
$<Server\ Name = "cpu"\ Quantity = "1"\ SchedulingPolicy = "PS"/>$
$<SinkNode\ Name = "OUT"/>$
$</Node>$
$<Workload>$
$<OpenWorkload\ WorkloadName = "withdraw"\ ArrivalRate = "1.0"\ ArrivesAt = "sourcen1"\ DepartsAt = "OUT">$
$<Transit\ To = "cpu"\ Probability = "1.0"/>$
$</OpenWorkload>$
$<OpenWorkload\ WorkloadName = "getbala"\ ArrivalRate = "1.0"\ ArrivesAt = "sourcen2"\ DepartsAt = "OUT">$
$<Transit\ To = "cpu"\ Probability = "1.0"/>$
$</OpenWorkload>$
$</Workload>$
$<Arc\ FromNode = "sourcen2"\ ToNode = "cpu"/>$
$<Arc\ FromNode = "sourcen1"\ ToNode = "cpu"/>$

	ClosedWorkload			OpenWorkload		
	NumberOfJobs	ThinkDevice	ThinkTime	ArrivalRate	ArrivalsAt	DepartsAt
Withdraw				1.0	Sourcen1	OUT
Getbala				1.0	Sourcen2	OUT

Table 8: Workload Table Informationle for the ATM example

```
< Arc FromNode = "disks" ToNode = "cpu"/ >
< Arc FromNode = "atm" ToNode = "cpu"/ >
< Arc FromNode = "cpu" ToNode = "disks"/ >
< Arc FromNode = "cpu" ToNode = "atm"/ >
< Arc FromNode = "cpu" ToNode = "OUT"/ >
< ServiceRequest >
< TimeServiceRequest WorkloadName = "withdraw" ServerID = "disks" ServiceTime = "0.05" NumberOfVisits =
"8" >
< Transit To = "cpu" Probability = "1.0"/ >
< /TimeServiceRequest >
< TimeServiceRequest WorkloadName = "getbala" ServerID = "disks" ServiceTime = "0.05" NumberOfVisits =
"3" >
< Transit To = "cpu" Probability = "1.0"/ >
< /TimeServiceRequest >
< TimeServiceRequest WorkloadName = "withdraw" ServerID = "atm" ServiceTime = "1.0" NumberOfVisits =
"11" >
< Transit To = "cpu" Probability = "1.0"/ >
< /TimeServiceRequest >
< TimeServiceRequest WorkloadName = "getbala" ServerID = "atm" ServiceTime = "1.0" NumberOfVisits = "6" >
< Transit To = "cpu" Probability = "1.0"/ >
< /TimeServiceRequest >
< TimeServiceRequest WorkloadName = "withdraw" ServerID = "cpu" ServiceTime = "3.15E−4" NumberOfVisits =
"18" >
< Transit To = "disks" Probability = "0.4"/ >
< Transit To = "atm" Probability = "0.55"/ >
< Transit To = "OUT" Probability = "0.05"/ >
< /TimeServiceRequest >
< TimeServiceRequest WorkloadName = "getbala" ServerID = "cpu" ServiceTime = "2.5E − 4" NumberOfVisits =
"8" >
< Transit To = "disks" Probability = "0.3"/ >
< Transit To = "atm" Probability = "0.6"/ >
< Transit To = "OUT" Probability = "0.1"/ >
< /TimeServiceRequest >
< /ServiceRequest >
< /QueueingNetworkResults >
```

This file can be easily validated against the PMIF 2.0 XML Schema (the complete schema definition may be seen at http://www.perfeng.com/pmif/pmifschema.xsd)

6 CONCLUSIONS AND FUTURE WORK

A Performance Model Interchange Format (PMIF) provides a mechanism whereby system model information can be interchanged between different performance modeling tools. The exporting and importing tools can either support the PMIF or provide an interface to read/write model specifications from/to a PMIF file. This paper describes the non-trivial process carried out in order to implement the Qnap exporting mechanism to PMIF. This process requires the lexical and syntactical analysis of the Qnap input file in order to make possible the translation to the PMIF format.

A separate research project is also developing the complete meta-model for performance results and the XML schema for it. Future work is then the development of the Qnap exporting mechanism for the results. Moreover, the consequences of other possible PMIF modifications (as for instance adding solving instructions) on the the exporting mechanism should also be addressed in the near future.

So far, we have developed PMIF export mechanism prototypes for Qnap and for *SPE·ED* and the importing mechanism for Qnap (details of the last two can be found in [8]). We would like to use this experience and the prototypes to develop some common routines (i.e. an API) that would make it easier for a tool developer to implement an internal interface to PMIF.

Appendix: Compiler Techniques and Tools

A programming language can be defined by describing the aspect of its programs (the *syntax* of the language) and the meaning of its programs (the *semantics* of the language) [1]. In order to represent the syntax of the language, a well known notation is used called context-free grammars or BNF (for Backus-Naur Form). Besides specifying the syntax of a language, a context-free grammar can be used to help guide the translation of programs, since a grammar naturally describes the hierarchic structure of most constructions of programming languages.

In the lexical analysis (or scanning) the stream of characters making up the source program is read from left-to-right and grouped into tokens that are sequences of characters having a collective meaning. Each token is treated as a unique entity. The blanks separating the the characters of these tokens would normally be eliminated during lexical analysis.

In the syntax analysis (or parsing) the characters or tokens are grouped hierarchically into nested collections with collective meaning. In other words, syntax analysis involves grouping the tokens of the source program into grammatical phrases that are used to synthesize output. Usually, the grammatical phrases of a source program are represented by a parse tree. A parse tree describes the syntactic structure of the input.

For the transformation of a Qnap file into a file in PMIF format we need a lexical analyzer and a syntax analyzer. There exist software tools that make it possible to easily create such analyzers. Some examples are *JLex* [11] and *JavaCup* [4] which use Java technologies.

JLex makes possible the generation of lexical analyzers. The user only needs to write a specification file (.lex) where the regular expressions that the analyzer needs to find are defined. Using this file, *JLex* quickly builds a Java application (a lexical analyzer) that can analyze files composed of chains of characters.

The resulting lexical analyzer searches the file to find strings that fit the regular expressions contained in the specification file (.lex). The regular expressions are defined in the .lex file together with the action to carry out when these expressions are found by the lexical analyzer. A regular expression represents a set of chains and the action is a set of orders that will be executed when the lexical analyzer finds a chain that fits the regular expression.

JavaCUP is an utility for generating syntax analyzers. The user needs to provide *JavaCUP* with an input file (.cup) where the grammar is specified, and then *JavaCUP* generates a java syntax analyzer. This analyzer can check whether an input text file adheres to the specified grammar.

The grammar specification file (.cup) includes the grammar specified in terms of terminal symbols, non terminal symbols, and a set of production rules. It can also include java code for executing specific actions during the analysis.

The syntax analyzer and the lexical analyzer work together. The syntax analyzer asks the lexical analyzer for the tokens (terminal symbols) it encounters.

Fig. 3 summarizes this process.

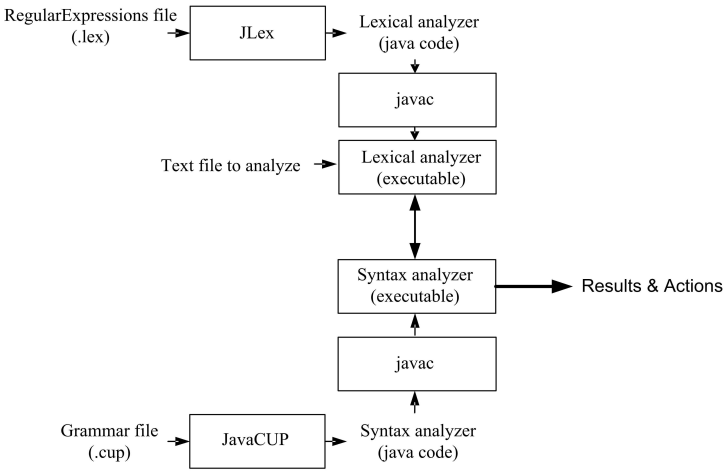


Figure 3: Lexical and Syntax Analysis Process

References

[1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers Principles Techniques and Tools*. Addison-Wesley, 1986.

[2] M. Potier, D.; Veran. Qnap2: A portable environment for queueing systems modelling. In D. Potier, editor, *I International Conference on Modeling Techniques and Tools for Performance Analysis (Paris, May 1984)*, pages 25–63. North Holland, May 1985.

[3] J. Rossello, C.M. Llado, R. Puigjaner, and C.U. Smith. A web service for solving queueing networks models using pmif. Tecnical report. To be plublished in Proceedings of the Fith International Workshop of Software and Performance, July 2005.

[4] C. Scott Ananian Scott Hudson, Frank Flannery. Cup parser generator for java. www.cs.princeton.edu/ appel/modern/java/CUP/.

[5] Simulog. Qnap modelling tool. www.simulog.fr.

[6] Simulog. Modline 2.0 qnap2 9.3: Reference manual, 1996.

[7] C.U. Smith and C.M. Llado. Performance model interchange format (pmif 2.0): Xml definition and implementation. In *Proc. of the First International Conference on the Quantitative Evaluation of Systems*, pages 38–47, September 2004.

[8] C.U. Smith and C.M. Llado. Performance model interchange format (pmif 2.0): Xml definition and implementation. tecnical report. www.perfeng.com/paperndx.htm, April 2004.

[9] C.U. Smith and L.G. Williams. Panel presentation: A performance model interchange format. In *Proc. of the International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, 1995.

[10] C.U. Smith and L.G. Williams. A performance model inter-change format. *Journal of Systems and Software*, 49(1), 1999.

[11] Elliot Berk.Princeton University. Jlex: A lexical analyzer generator for java(tm). www.cs.princeton.edu/ appel/modern/java/JLex/.

[12] w3c. World wide web consortium. www.w3c.org, 2001.

An Infrastructure for Implementing Compilers for Concurrent Abstract State Machine Languages

Kristian Magnani, Mariza A. S. Bigonha, Roberto S. Bigonha, Fabíola F. Oliveira
Universidade Federal de Minas Gerais, Departamento de Ciência da Computação,
Belo Horizonte, Brazil, 31.270-901
{kristian, mariza, bigonha, fabiola}@dcc.ufmg.br

and

Vladimir O. Di Iorio
Universidade Federal de Viçosa, Departamento de Informática,
Viçosa, Brazil, 36570-000
vladimir@dpi.ufv.br

Abstract

This paper proposes a new approach for concurrency in abstract state machines (ASM) and presents the MIR Architecture, an infrastructure designed to serve as basis for abstract state machine compilers. This new concurrency model is based on Lamport's concept of delayed knowledge. Besides the implementation of this new approach, the proposed infrastructure also allows the implementation of optimizations specially designed in order to address specific ASM optimization opportunities.

Keywords: MultiAgent Systems, ASM, Distributed System, Intermediate Representation Language.

1 Introduction

Abstract State Machines (ASM) is a formal semantic method introduced by Yuri Gurevich in order to provide operational semantic for algorithms [11, 12]. This paper describes a new model for specifying concurrent systems based on ASM and proposes an infrastructure, called MIR, to implement ASM-like languages. This infrastructure provides two important features:

- It implements the concurrency model proposed in this paper. This model is based on Lamport's concept of distributed systems [15] and delayed knowledge.
- It is also designed to allow some optimizations to be performed over ASM specifications [20, 24, 25]. These optimizations can be those specific for the ASM model, and do not overlap with the customary code optimizations.

According to Gurevich, ASM are abstract machines whose states are algebraic structures, which can be viewed as abstract memories. Functions in the algebra and their arguments are locations of memory and the values of the functions are their contents [4]. Some basic concepts related to the ASM model are introduced below.

Vocabulary and states: a *vocabulary* or *signature* is a finite collection of function names, each with a fixed arity. A *state* is a nonempty set called the *superuniverse*, together with interpretations of the function names.

Transition Rules: A transition rule resembles a program written in an imperative language. In a transition rule there is no iteration command,because it is not necessary due to the intrinsic cyclic execution of an ASM. Given an *initial state*, the transition rule is applied to it, leading to a new state, over which the transition rule is applied again. This process repeats over and over, until no modifications are observed in the state. Although the vocabulary stays unchanged along the execution, its interpretation is modified by the transition rule, from state to state. The execution of ASM transition rules produces update sets, which is used to obtain the new state. Transition rules are *update rule*, *block constructor* and *conditional constructor*. An *update rule* is an expression $f(\bar{t})$:= t_0, where f is the name of a function, $\bar{t}$ is a tuple of terms whose length equals the arity of f and t_0 is another term. Terms have no free variables and are recursively built using names of distinct elements of the superuniverse and the application of function names to other terms. The effect of the execution of such a rule is that, in the *next* state, the value of the function f at the point $\bar{t}$ is the value denotated by t_0, evaluated at the *current* state. A *conditional constructor* is an expression with the following format: **if** g_0 **then** R_0 **elseif** g_1 **then** R_1 ... **elseif** g_k **then** R_k **endif** The semantics is: the rule R_i, $0 \leq i \leq k$, will be executed if the boolean terms $g_0, ..., g_{i-1}$ evaluate to *false* and g_i evaluates to *true*, in a given state. A *block constructor* is a set of rules $R_0, R_1, \ldots, R_k$. The effect of the execution of a block is the execution of all its components at the same time.

Programs and Runs: a program is a transition rule. A *run* is a sequence of states. Each state is generated by firing an update set in the previous state.

The original Gurevich's ASM also contemplates the definition of concurrent systems. In that case, agents that do not make use of a common resource can even be executed in parallel. However, the coherence condition requires that the moves of the agents are ordered. Each agent acts over a world that, in the very moment of its action, can be changed exclusively by it. It is as if the execution of a transition rule is always performed inside a critical section. This approach leads to a simpler, more tractable model, but it does not reflect the reality of distributed systems.

According to Lamport [15], a distributed system may be defined as a collection of processes that are somehow separated and communicate with each other by exchanging messages. If inside a system the message transmission delay is not negligible comparing to the time between events in a single process, then such a system can be considered distributed.

One of the remarkable features of a distributed system is that it is not always possible to say if an event will occur before or after another one. So, it is possible to get some non-predictable behaviour from a distributed system, regarding that the order of events is not predictable. It is the responsibility of the designer to use the available synchronization mechanisms in order to avoid such undesirable, anomalous behaviour.

Section 3.2 presents the proposed model which implements these ideas. Of course, the original Gurevich's ASM is a more general model, so it is possible to describe the one proposed in this paper in terms of the original model. However, we believe that introducing these semantics, as done in this work, could simplify the definition of systems in some contexts.

2 Related Work

Del Castillo presents in [7] a definition of an *evolving algebra abstract machine* (EAM) as a platform for developing ASM tools and [6] introduces an implementation called ASM-Workbench. The ASM-Workbench [5, 6] describes a system that is able to transform an ASM specification to a C++ program. The specification language is called ASM-SL, and it is a typed ASM specification language based in functional programming language ML. The ASM-Workbench is an important implementation. It is also given a formal definition of the EAM ground model in terms of a universal ASM. [8] (apud [7]) performs a description of a functional interpreter for ASM, with applications for functional programming languages. Some extensions to the language of ASM are proposed, as well. [13] (apud [7]) presents a Prolog interpreter for ASM specifications that are made in a particular language.

The AsmGofer is an ASM programming environment presented by Schmidt [22, 21], which extends the Gofer functional language. It provides an interpreter, and therefore it is not so fast as a compiled specification could be. On the other hand, it is useful in order to build prototypes.

Anlauff presents in [1] the XASM, an ASM language, together with a compiler for it. A formal definition of the language is given by Kutter in [14].

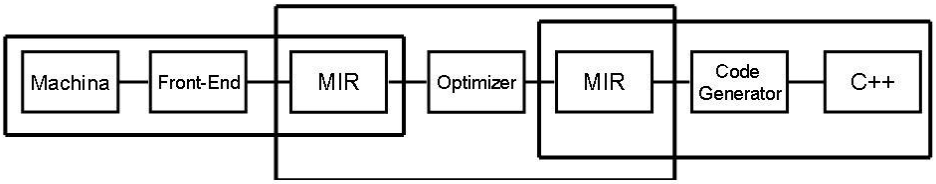


Figure 1: The Machĭna Project.

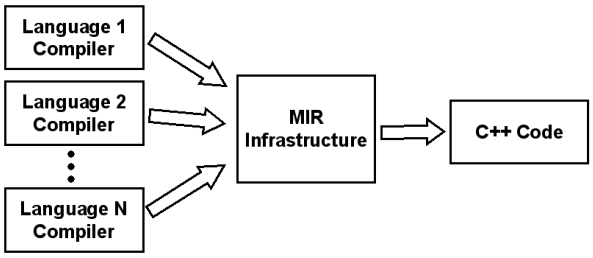


Figure 2: The Context of MIR.

The current AsmL version of Gurevich is AsmL 2, also known as AsmL for the Microsoft .NET, and it can be found at [2]. It does not provide multi-agent mechanisms. On the other hand, it benefits from the vast library of the Microsoft .NET platform.

Finally, Visser have developed the EvADE compiler [27], which implements an optimization by means of common sub-expression elimination. However, this one does not belong exclusively to the ASM model.

All these systems are concerned with only a few optimizations, and none of them addresses the concurrency issue nor mention an intermediate representation with features for disttributed ASM. The infrastructure presented by this paper aims to properly address the concurrency issue, using an intermediate representation language, and it also provides an optimization environment where specific ASM optimizations can be plugged in order to produce efficient code. The infrastructure proposed and how these results are achieved are explained in the sequel.

3 The MIR Architecture

The idea of developing a general infrastructure for compiling *concurrent* ASM specifications arised from the purpose of bringing this powerful semantics modeling methodology to the world of distributed systems. It would be helpful to have a general infrastructure available, which could be the basis of compilers aiming at different ASM oriented languages. As it provides the concurrent and parallel execution capabilities, such an infrastructure would be useful to implement parallel algorithms, at the same time providing a formal semantics model over which one could prove or deduce properties. The MIR architecture was designed in order to answer these needs. MIR stands for Machĭna Intermediate Representation. It was originally used as an intermediate representation for the Machĭna language under the Machĭna project. The MIR project has grown up and nowadays it serves as the basis of compilers for concurrent ASM oriented languages.

The Machĭna project, illustrated in Figure 1, is composed of three distinct parts. The first one comprises the Machĭna front-end, which translates Machĭna program to MIR, the intermediate representation language. The second part receives a MIR file, optimizes it, considering the optimizations specially tailored for the ASM model and produces as output a MIR file optimized. The third part translates from MIR to C++.This paper addresses the design and implementation of MIR. For details of the Machĭna project refers to [3, 16, 17, 19].

The main feature of the MIR Infrastructure is to produce C++ code from a MIR specification and thus allowing rapid development for ASM compilers. Figure 2 shows MIR usage context.

Another remark worth mentioning about MIR is that it was designed to be easily *optimized*, as shortly introduced in Section 3.5. These optimizations go beyond those who are normally performed by the C++ compiler. Rather, they are related to the ASM model. For details, see [18, 20].

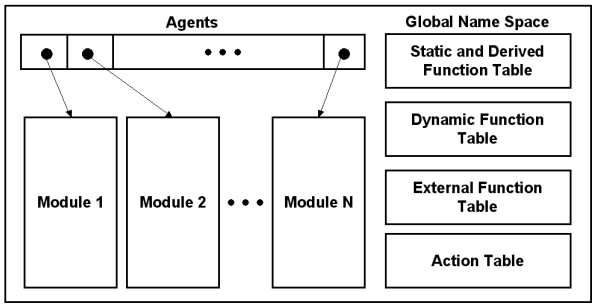


Figure 3: The MIR Architecture.

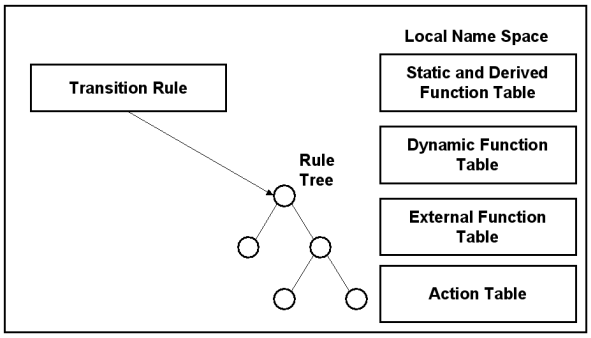


Figure 4: A Module of MIR.

MIR Infrastructure means all the classes and software components that compose the infrastructure presented in this paper and its implementation. The expression *MIR Architecture* refers to the general structure of an ASM specification using the MIR Infrastructure .

3.1 Agents, Modules and Other Elements

A MIR specification is basically composed by a set of *agents*, each of them of a given type, and a *common* Global Name Space, which is accessible by each agent belonging to that MIR specification. This picture is depicted in Figure 3. The Global Name Space is defined as tables for *static*, *derived*, *external* and *dynamic functions*, and for the *actions*, as well. Static functions are those which can not have values in points of their domain changed by update rules. These functions are defined by parameterized expressions, and remain unchanged during the whole execution of the MIR architecture. It is not allowed to call a dynamic function inside its definition, as well. A derived function is similar to a static function, except that it may call dynamic functions. Dynamic functions can have values in points of their domain changed or even defined by update rules. External functions are those defined outside the MIR architecture definition, and only their signatures and return types are known. They are useful to model interaction with the environment. Finally, actions are the abstraction of agents. They allow the implementation of the notion of *submachine*, as pointed out by Tirelo et alii [24]. Static and derived functions are grouped together, because their definitions are closely related.

The type of an agent is a *module*. Figure 4 presents the structure of a module in MIR. A module contains a *transition rule* augmented with some support structures: the *update lists* and the Local Name Space. Like the global one, the Local Name Space consists of the tables for static, derived, external and dynamic functions, and for the actions, as well.

The transition rule is a tree of rules. The nodes of such a tree are given by the simple and composite rules of the MIR Architecture. Simple rules appears always in the leaves of a rule tree, while composite rules are used in order to build rules from other ones. Simple rules are:

- *update* rule, which changes the value of a dynamic function at a specific point;

- *create* rule, which creates a new agent and starts its execution;
- *destroy* rule, which deletes an agent;
- *stop* rule, which indicates that the execution of the rule must be halted;
- *return* rule, used inside actions to indicate that the execution must return to the calling rule;
- *action call* rule, which starts an action as a submachine;

The composite rules of the MIR Architecture are:

- *conditional* rule, the traditional *if-then-else* rule;
- *forall* rule, which applies a rule to *all* the elements of a set;
- *choose* rule, which applies a rule to *a randomly chosen* element of a set;
- *let* rule, which allows *ad-hoc* expressions definitions;
- *case* rule, which executes a rule according to the *value* of an expression;
- *with* rule, which executes a rule according to the *type* of an expression;
- and the *block* rule, used in order to define a rule that is the union of many rules.

The Static and Derived Function Table is a list whose entries have three components: Function Name, Function Type and Function Definition, as depicted in Figure 5. The function type is a tree of types, whose root is possibly a functional type node, except when the function is nullary. A Type Tree is a tree whose nodes are either basic types or type constructors. High order functions are not allowed. The Function Definition is a tree, as well, and it can be recursively constructed from basic expressions and expressions constructors. The definition of each of the type and expression nodes are beyond the scope of this paper, and the reader is referred to [18] in order to get more details about them.

The entries of the Dynamic Function Table have also three components, but its third component, the Function Definition, now leads to a dynamic mapping between points in the function domain and their values. The Dynamic Function Table is illustrated in Figure 5.

As external functions are defined outside the MIR architecture definition, the entries of the external functions table have just two components: the Function Name and the Function Type. External functions are written in C, and the communication protocol and the parameter mapping policy are defined in Section 3.3. The External Function Table is depicted in Figure 5.

The Action Table represents the table of agent abstractions, and its entries are pairs whose first component is the Action Name and the second component is the Rule Tree. This table is presented in Figure 5.

3.2 The MIR Approach for Concurrency

Although the original definition for concurrent ASM programs, as presented in [11], allows the definition of concurrent systems, it does not offer facilities for the definition of multiagent systems. In the Gurevich's model, two or more agents are not allowed to execute their transition rules simultaneously if they make use of a common resource, because conflicts in using a shared resource are resolved by partially ordering the conflicting moves [4].

According to Zambonelli [29], an *agent* is a software entity that exhibits *autonomy*, *situatedness* and *proactivity*. The concept of agent used here pursuits these features in the following sense:

Autonomy: there is a transition rule associated with each agent, giving it an autonomous and independent behaviour from other agents.

Situatedness: every agent is immersed in a common global context where they can interact among each other, which are referenced by their names or by the special word *self*[1].

Proactivity: an agent has the freedom to call actions over other agents.

[1]The interpretation of *self* is the agent itself in whose transition rule this word is found.

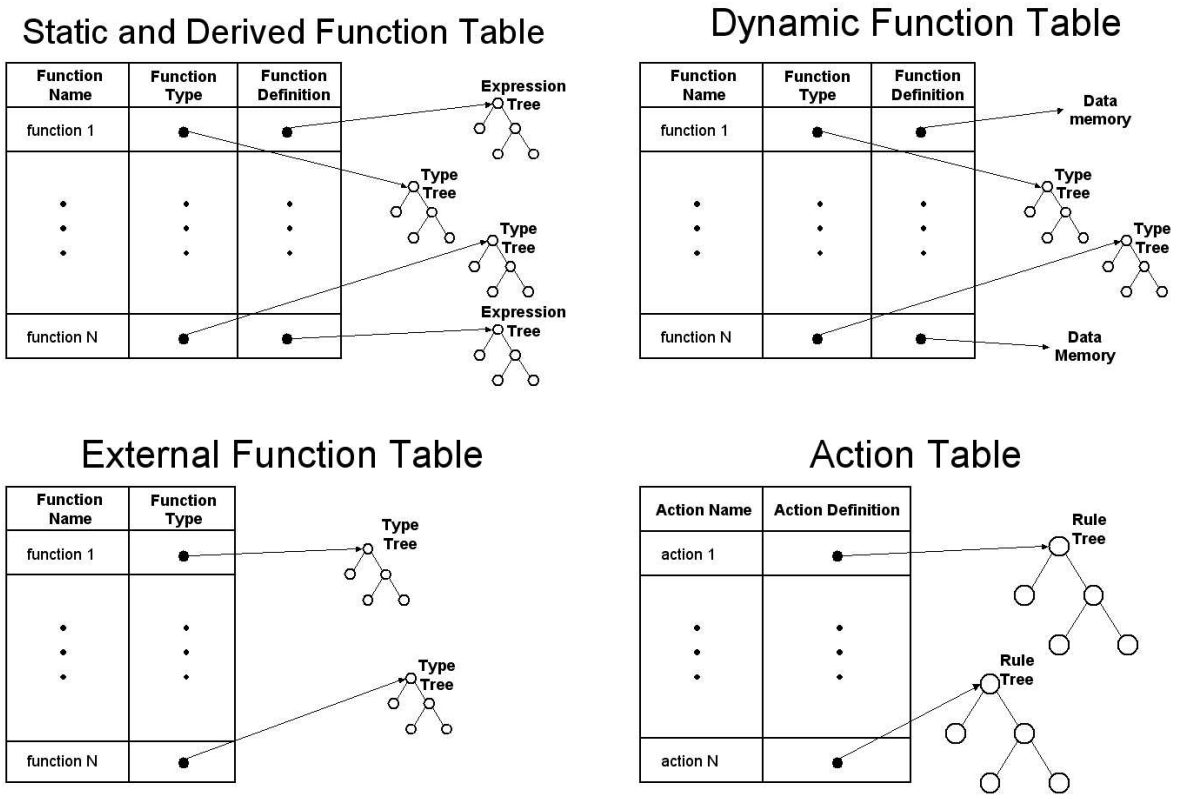


Figure 5: The Structures of the Tables.

Agents also interact with each other through cooperation, coordination or negotiation. This kind of interaction is called *sociality*.

This section presents a new approach for concurrency in ASM in order to cope with real parallel systems. The main features of this new approach are the possibility of simultaneous moves by different agents and the concept of "delayed knowledge" [15], meaning that agents may store copies of the global state which may not always be kept up to date. The definition of this approach is provided here in an informal but detailed fashion. A formal definition of these ideas are under preparation.

Our intention is to model systems with agents of different speed of execution. When the update set of an active agent is processed, it means that the agent has had enough time to execute an iteration of the associated transition rule. This time can differ from agent to agent, leading to an unpredictable delay between the moment some changes are performed by one agent and the moment when these changes are perceived by another agent. In a distributed system, the delay may be also associated with the time for information transmission.

The proposed model does not provide any primitive for communication between agents. The model is general enough to allow inconsistent simultaneous updates produced by different agents. All communication must be explicitly programmed by the designer of an ASM concurrent specification. For example, suppose a set of agents in a *distributed* system, meaning that all communication is handled via message exchange. The designer of the system may represent communication channels by ASM functions which are updated by a *sender* agent and read by a *receiver* agent, avoiding inconsistent updates. Many ASM-based languages [26, 2] offer mechanisms for hiding information and visibility control, and also for the definition of abstractions for ASM rules. These concepts may be used by the designer to ensure that all communication will be executed using only the defined channels, allowing an agent to update only the ASM functions related to its output channel.

In the MIR Architecture, the scope of a function or action may be declared either *local* or *global*. Local declaration specifies the elements that belong to a specific agent, and therefore these elements are accessible

just inside that agent. Conversely, a globally declared function or action can be accessed by each agent at execution in the context of a MIR specification. Global elements are declared at the top-most level, namely, the MIR specification outside the modules. Name conflicts between elements of both scopes are not allowed.

Every agent in a MIR specification is executed concurrently, and each of them has local copies of the global dynamic functions it makes use of. The local copies are used by the transition rule of the agent, and occasionally a synchronization window opens and the local copies and their global correspondents are updated. As argued by Lamport in [15], in a concurrent system, it is sometimes impossible to say in advance which one of two events will occur first, and therefore the relation "happened before" is only a partial ordering of the events of the system. This is particulary true for the event of synchronization in the MIR approach for concurrency, since the synchronization is made by each agent at the end of the execution of its transition rule, and it is not possible to assure that every agent will be going to finish it at the same time.

The execution of an agent transition rule produces three types of update lists, namely: the *local* updates, the *import* updates and the *export* updates. These update lists are maintained inside the MIR module, and are employed in the proper situation.

The local updates are related to the update rules that act over locally defined dynamic functions. They are fired at the end of every execution of the transition rule. The entries of the local update list are cleaned up after they are fired, and then the new execution of the transition rule will fill in it again. This gives the local update list a transient, dynamic nature. The import and export update lists, however, are associated with globally defined dynamic functions.

The import updates are fired every time the local copy of a global dynamic function needs to be refreshed. This happens at two occasions:

1. the import is done when the execution of an agent starts;
2. it is also done at the moment the values of dynamic functions of the agent are synchronized with the global name space.

The entries of the import update list of a module are all those globally defined functions used as *right* hand side values by the transition rule of the module. This list can be constructed from the analysis of the transition rule, and it remains unchanged during all the execution. The export updates take effect when the local copy of a global defined dynamic function is to be uploaded into its global counterpart, hence its local value becoming available to other agents which imports that dynamic function. The entries of the export update list of a module are all those updates of global dynamic functions used as *left* hand side values by the transition rule of the module.

According to Lamport [15], a distributed system may be defined as a collection of processes which are somehow separated, and they communicate with each other by exchanging messages. If inside a system the message transmission delay is not negligible comparing to the time between events in a single process, then such a system can be considered distributed. The concurrent MIR approach follows the above definition. More specifically, each agent performs a data exchange at the end of each iteration of its transition rule, and it is at this very moment that the agent updates its knowledge about the external world. Between two consecutive data synchronizations, the agent gets its transition rule executed, which may take an unbounded period of time. This time differs from agent to agent since it depends upon several factors, like the size of the rule, the parts of this rule that are executed in fact, processor speed and other hardware resources, and so on. Meanwhile, the external environment can be modified by other agents, but these changes are perceived by an arbitrary agent only at its next synchronization, and this time is not negligible. As it occurs with real life distributed systems, the perception of reality may differ depending on the moment of observation.

3.3 MIR Native Interface

MIR Architecture allows the use of *external functions*, also known as *oracle functions*, to be written as C++ *functions*. External functions are present in the ASM model since the beginning and they provide a way to specify interaction with the environment. As argumented by Gurevich in [11], an oracle function does not need to be consistent between different execution steps of a transition rule. But the oracle should be consistent at different uses in the same execution step of the transition rule. This effect can be achieved by caching the accesses of the external functions at the same iteration, and that is done in MIR Infrastructure. In order to get the external functions written in C++ to be called from the MIR specification properly, it is required that they obey some conventions. These conventions are called *MIR Native Interface*, or MNI, for

MIR Type	Correspondent C++ Type
boolean	`bool`
character	`char`
integer	`int`
real	`double`
string	`std::string`
$(T_1,\ldots,T_n)$	`struct`
list of T	`std::vector<`T`>`

Table 1: Type mapping in the MIR Native Interface. The `std::` prefix means that the element belongs to the *Standard Template Library* of C++ [23].

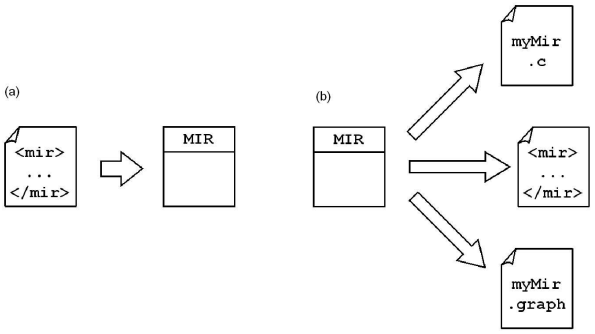


Figure 6: The features of the MIR implementation.

short, and they determine a common protocol upon which both the MIR specification and the C++ external function can rely.

In the external functions written in C++ it is not allowed the void return type, as they are indeed not procedures, but functions, and so some value is expected from them. The mapping from the types of MIR and the correspondent C++ types are presented in Table 1. These are the only types allowed both in the signature and as return type of the C++ functions used as external ones. Due to implementation restrictions, the type of a parameter can only be a basic type, a tuple, or a list of the allowed types.

3.4 Highlights of MIR Implementation

The implementation of MIR is heavily based on design patterns, particularly on the *visitor* design pattern. According to Gamma et alii. [9], a visitor is a design pattern that represents an operation to be performed on the elements of an object structure without changing the classes of the elements upon which it operates. This concept is directly applied in the implementation of the following operations.

Serialization: The MIR representation of an ASM specification can be saved in persistent store. This saving process is called *serialization*. MIR implementation allows its serialization through XML files, according to a specific format determined by a XSD (*XML Schema Definition*) [28]. This serialization may happen in both ways: it is possible to get a MIR object from a serialization file, as well as a MIR object can be serialized in such a file. The main advantages of this represention are: XML files are just plain text files, and so they can be edited according to the needs of the programmer; XML files are easily readable not just by humans, but also by machines, as it is relatively simple to build parsers for them, and even some libraries are available in order to automate this work; and finally, an XML representation is quite easy to produce from the hierarchical structure of the MIR architecture.

Compilation to C++ Code: The translation of MIR program to C++ code is implemented by means of a visitor. The generated code matches the C++ standards, so it is possible to compile it with a few changes into different machines and different operating systems. The choice of C++ as the target language allows

the generation of efficient, fast code, which is essential in many scenarios. The existence of several C++ compilers, some of them available without costs, frees the user of the MIR Infrastructure of being dependent upon specific compilers and vendors.

Direct Execution: The MIR implementation allows its direct execution. In other words, the MIR objects can be executed in an interpreted fashion, without the need of being converted to C++ code and then compiled. This is useful, for instance, in building step-by-step symbolic debbugers. The visitor that provides this feature can be viewed as a virtual machine for executing MIR specifications, a concept used nowadays and that offers some advantages, as providing an abstraction layers over different machines and operating systems.

Visualization: It is also possible to obtain a visual representation of a MIR specification through the generation of its description in the DOT language of the GraphViz software. The DOT language is a language designed for description of graph-like structures, and it is used in GraphViz software. This program and the definition of the DOT language can be found at [10].

3.5 MIR and Optimization

It is not enough to a modern compiler to get executable code; this code should be also *optimized*. For the target language of the MIR Infrastructure, namely, C++ code, there are several compilers that address this issue with efficiency. However, there are some optimization opportunities that are particular to the ASM model, and therefore they are not performed by the existing C++ compilers, as argued in [20] and in [25]. These opportunities are the ones we are interested in.

In order to address this special situation, it is under development a framework [17] that provides the proper environment to plugin specific MIR optimizations, as depicted in Figure 7. The optimizations are easily added or removed as plugins, thus facilitating the development of new optimizations algorithms.

As examples of such ASM-related optimizations, we point out the Update Scheduling Optimization and the Jump Optimization, taken from [19, 20, 25]. Some brief comments about them are made in the sequel.

Update Scheduling Optimization: Let $B = U_1, U_2$ be a block where U_1 and U_2 are updates, so that U_1 is `y:=z` and U_2 is `x:=y`. According to the semantics of ASM, these updates could not be directly converted to a sequence of assignments in C++ code. Instead, the updates must be compiled into code that inserts entries in the list of updates to be processed at the end of the iteration of the rule. Unfortunately, this can be time consuming, as the operations involved may have high computational costs. However, some intructions, like U_2, could be performed immediately without the loss of the model features. The order in which the elements of a block are compiled can result in a greater or smaller number of direct updates. For instance, if U_1 and U_2 are commuted, no insertion in the update list would be necessary, and both updates could be carried out directly in place. The optimization proposed schedules the rules inside a block in order to minimize the number of insertion in update list.

Jump Optimization: A transition rule R must be compiled into an infinite loop of the form `L:` R `; goto L;` . In the case of conditional rules, like `if` g `then` R_1 `else` R_2, it is compiled into `L: if (`g`)` R_1`; else` R_2`; goto L;`. Now suppose that R_2 does not update any dynamic function used in g, and no external function call can be found in g, as well. In this case, once R_2 executed, it is not necessary to get back to `L`, as the reevaluation of g produces the same value. An optimized code could be like `L: if (`g`)` $\{R_1\}$`; else` $\{$`L2:` R_2`; goto L2; }` `goto L;`. This optimization detects possibilities like the one above, producing code that results in more efficient execution.

4 Conclusions

After a brief review of the ASM model, this paper has presented the MIR Architecture which consists of an infrastructure for developing concurrent ASM oriented languages. The most important contributions of the proposed infrastructure are: (1) It can be used in order to implement a whole family of languages targeting the ASM model. (2) The concurrent approach and the ASM modelling distributed agents are new, and it can be used to precisely describe distributed algorithms in ASM. (3) Optimizations can be plugged in, allowing

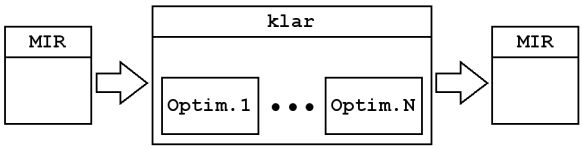


Figure 7: The optimization process of a MIR specification by using the kℓar framework.

enhancements of the generated code. Moreover, these pluggins can also be used to other types of program manipulation, e.g. refactoring, given that the necessary information is provided.

Future work includes the formal definition of the concurrent ASM model used in the infrastructure and the demonstration that important properties of disttributed systems hold for it. Some examples of these properties include the absence of starvation and the possibility of mutual exclusion.

References

[1] M. Anlauff. XASM – An Extensible, Component-Based Abstract State Machines Language. In Y. Gurevich and P. Kutter and M. Odersky and L. Thiele, editor, *Abstract State Machines: Theory and Applications*, volume 1912 of *LNCS*, pages 69–90. Springer-Verlag, 2000.

[2] asml. The AsmL 2 for Microsoft .NET Homepage: http://research.microsoft.com/fse/asml/. Consulted in 20th April 2005.

[3] Roberto S. Bigonha, Fábio Tirelo, Marcelo A. Maia, Marcelo Silva, Marco Túlio O. Valente, Mariza A. S. Bigonha, and Vladimir O. Di Iorio. Projeto Machina. Technical Report LLP007/99, Universidade Federal de Minas Gerais, Julho 1999.

[4] E. Börger and R. Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer-Verlag, 2003.

[5] G. Del Castillo. The ASM Workbench: an Open and Extensible Tool Environment for Abstract State Machine. In *28th Annual Conference of the German Society of Computer Science*, 1998.

[6] G. Del Castillo. *The ASM Workbench: A Tool Environment for Computer-Aided Analysis and Validation of Abstract State Machine Models*. PhD thesis, Universität Paderborn, 2000.

[7] G. Del Castillo, I. Durdanović, and U. Glässer. An Evolving Algebra Abstract Machine. In H. Kleine Büning, editor, Proceedings of the Annual Conference of the European Association for Computer Science Logic (CSL'95), volume 1092 of *LNCS*, pages 191–214. Springer, 1996.

[8] D. Diesen. *Specifying Algorithms Using Evolving Algebra. Implementation of Functional Programming Languages*. Dr. scient. degree thesis, Dept. of Informatics, University of Oslo, Norway, March 1995.

[9] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[10] GraphViz. The GraphViz Homepage: www.graphviz.org, 2005. Consulted in February 2005.

[11] Y. Gurevich. Evolving Algebras 1993: Lipari Guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1995.

[12] Yuri Gurevich. Evolving algebras: An attempt to discover semantics, 1991.

[13] A. M. Kappel. Executable Specifications Based on Dynamic Algebras. In A. Voronkov, editor, *Logic Programming and Automated Reasoning*, volume 698 of *Lecture Notes in Artificial Intelligence*, pages 229–240. Springer, 1993.

[14] P. Kutter. The Formal Definition of Anlauff's eXtensible Abstract State Machines. TIK-Report 136, Swiss Federal Institute of Technology (ETH) Zurich, June 2002.

[15] Leslie Lamport. Time, Clocks and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565, 1978.

[16] Mário Celso Candian Lobato. Proposta de Dissertação: Um Arcabouço para Compilação de Linguagens de Especificação ASM, 2005.

[17] Kristian Magnani. Proposta de Dissertação: kℓar - Um Arcabouço para Otimizações em Máquinas de Estado Abstratas, 2005.

[18] F. Oliveira, K. Magnani, M. Bigonha, and R. Bigonha. MIR: Machĭna Intermediate Representation. Technical Report RT001/04, Laboratório de Linguagens de Programação - Departamento de Ciência da Computação - Universidade Federal de Minas Gerais, 2004.

[19] Fabíola F. Oliveira. Proposta de Tese: Otimizações em Máquinas de Estado Abstratas, 2004.

[20] Fabíola F. Oliveira, Roberto S. Bigonha, and Mariza A. S. Bigonha. Otimização de Código em Ambiente de Semântica Formal Executável Baseado em ASM. *Proceedings of 8th Brazilian Symposium on Programming Languages*, pages 172–185, May 2004.

[21] J. Schmidt. Executing ASm Specifications with AsmGofer, 1999.

[22] J. Schmidt. Introduction to AsmGofer, 2001.

[23] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, third edition, 2000.

[24] Fábio Tirelo. Uma ferramenta para execução de um sistema dinâmico discreto baseado em Álgebras evolutivas. Master's thesis, Universidade Federal de Minas Gerais, 2000.

[25] Fábio Tirelo and Roberto S. Bigonha. Técnicas de Otimização de Programas Baseados em Máquinas de Estado Abstratas. *Proceedings of 4th Brazilian Symposium on Programming Languages*, pages 144–157, 2000.

[26] Fábio Tirelo, Marcelo A. Maia, Vladimir O. Di Iorio, and Roberto S. Bigonha. Projeto Machina: A linguagem de especificação ASM. Technical Report LLP008/99, Universidade Federal de Minas Gerais, Julho 1999.

[27] J. Visser. Evolving algebras. Master's thesis, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Zuidplantsoen 4, 2628 BZ Delft, The Netherlands, 1996.

[28] Priscilla Walmsley. *Definitive XML Schema*. Prentice Hall PTR, 2001.

[29] Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.

Evolución de la Selección Dinámica de Métodos en Java*

Víctor M. Gulías[1], David Cabrero[2], Carlos Abalde[1], Alberto Valderruten[2]
Grupo MADS, Departamento de Computación, Universidade da Coruña
Campus de Elviña s/n. 15071, A Coruña, España
[1]{gulias, carlos}@lfcia.org, [2]{cabrero, valderruten}@udc.es

Abstract

The static nature of the Java overloaded method resolution process is the reason for some design and implementation misunderstandings and problems. In this paper is presented a discussion about some of those questions.

The problem is presented and abstracted from a simple motivation scenario. Next, it is explored the relation with the well known Visitor design pattern. In both cases is found that the static method overload support available in the Java programming language is the reason of many implementation limitations.

Some problem workarounds are proposed and presented in detail: from a first and basic approach to a generic introspection-based technique. Finally, based on a real life Java application, is presented a similar scenario where neither of the workarounds are useful and where the limitations imposed by Java can not be avoided in an elegant way.

Keywords: Java, Design patterns, Method overload

Resumen

La naturaleza estática de la resolución de la sobrecarga de métodos en Java es motivo de problemas y confusiones a la hora de diseñar e implementar programas en dicho lenguaje. Es este artículo presentamos algunas reflexiones sobre estas cuestiones.

Presentamos y abstraemos el problema a partir de un escenario simple. A continuación, exploramos la relación con el conocido patrón de diseño del Visitante. En ambos casos encontramos que la resolución estática de la sobrecarga de métodos en Java es la razón de varias limitaciones en la implementación.

Proponemos varias alternativas, explicadas en detalle: desde una primera aproximación básica hasta una técnica genérica basada en introspección. Finalmente, basándonos en una aplicación Java del mundo real, se presenta un escenario similar donde ninguna de las alternativas resulta útil y no es posible evitar las limitaciones impuestas por Java de manera elegante.

Palabras claves: Java, Patrones de diseño, Sobrecarga de métodos

1 Introducción

Cuando David, una persona experimentada que ha trabajado durante años con diferentes paradigmas de programación (orientación a objetos, lógica, funcional, etc.), me mostró su aproximación para una de las prácticas de nuestra asignatura de Diseño de Sistemas Informáticos, me sentí confuso. Entendía su idea, pero sabía que esa solución no funcionaría con el algoritmo de invocación de métodos de Java. En concreto, el problema sería la naturaleza estática del proceso de resolución de métodos sobrecargados.

En este artículo presentamos una reflexión sobre algunos de los problemas causados por la sobrecarga estática de métodos en Java. También se incluye la relación con el conocido patrón de diseño Visitante [3] y una técnica genérica basada en instrospección [1] para resolver parcialmente el problema.

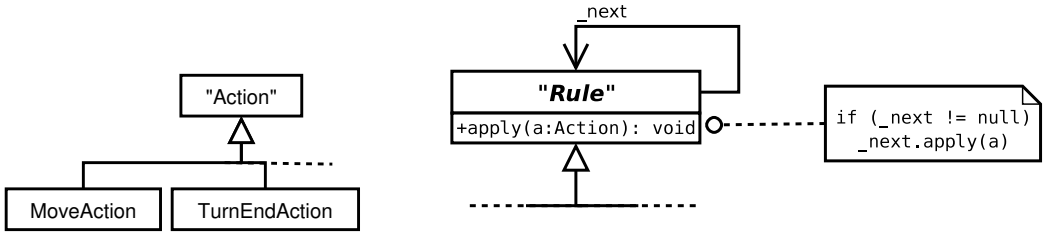
*Trabajo parcialmente subvencionado por el Ministerio de Ciencia y Tecnología español (proyecto TIC 2002-02859).

El artículo está estructurado de la siguiente manera: en primer lugar, se realiza una introducción y abstracción del problema motivador. A continuación la sección 3 presenta el problema en relación al patrón Visitante. La siguiente secciónestá dedicada a establecer algunas soluciones: desde una primera aproximación básica hasta una solución más genérica basada en instrospección. La sección 5 explica un problema relacionado en una aplicación Java del mundo real donde ninguna de las aproximaciones propuestas es útil y las limitaciones impuestas por Java no son fácilmente solventables. Finalmente, se presentan las conclusiones.

2 Descripción del Problema

2.1 Visión General del Escenario

Se parte una visión general del escenario que causa esta discusión. Deseamos modelar un subconjunto de un juego de estrategia por turnos. En este juego, el usuario selecciona en cada turno un conjunto de *acciones* (mover una unidad, finalizar el turno, etc.). Estas acciones se procesan generando un nuevo estado del juego. Desde el punto de vista del diseño, las acciones se modelan como una jerarquía, tal y como se muestra en la figura 1(a).



(a) Jerarquía *Action* (b) Jerarquía *Rule*

Figure 1: Diagrama de clases genérico del juego

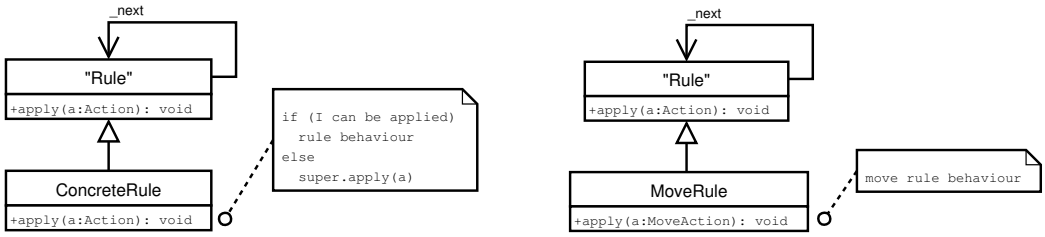
Por otra parte, el comportamiento interno del juego está gobernado por un conjunto de *reglas*. Estas reglas se buscan en un orden predefinido hasta que se encuentra una que encaja. Este comportamiento se modela usando el conocido patrón de diseño Cadena de Responsabilidad [3]: si la acción actual no encaja con la regla, se explora la siguiente regla en la cadena. Por tanto, como se muestra en la figura 1(b), el modelo de la jerarquía de reglas permite la definición de un comportamiento por defecto en la superclase de la cadena de responsabilidad. Aquí, el servicio definido por la cadena de responsabilidad (método `apply`) incluye la acción genérica (tipo `Action`) como parámetro.

2.2 Sobreescribiendo el Método apply

Las reglas concretas extienden la superclase `Rule` y deben decidir si pueden resolver el servicio (i.e., si la regla encaja). Si la regla es aplicable, entonces debe llevar a cabo su comportamiento específico. En caso contrario, debe llevar a cabo el comportamiento por defecto (i.e. el comportamiento modelado en la superclase `Rule`: delegar en la siguiente regla de la cadena).

Tal y como muestra la figura 2(a), el método `apply` ha sido reescrito en la clase derivada. La clase `ConcreteRule` tiene a todos los efectos un único método disponible: `apply(a:Action):void`.

Al contrario que en otros lenguajes de programación como C++, en Java, ante un método sobreescrito la selección del método a aplicar se realiza usando la *estrategia de máxima especifidad*: se selecciona siempre el método más específico de los disponibles en cada objeto concreto. Es decir, si el objeto A invoca el método `apply` sobre el objeto R ($R \in$ `ConcreteRule`), aunque A maneja R como una regla genérica (tipo `Rule`), siempre se aplicará la implementación del método `apply` [4]. Como se puede observar, la invocación de los métodos sobreescritos es de naturaleza dinámica. La selección de la implementación a invocar se realiza en tiempo de ejecución basándose en el tipo del objeto concreto, y obteniendo siempre la implementación



(a) Sobreescritura del método `apply` (b) Mejora del diseño

Figure 2: Sobrecarga del método `apply`

más específica disponible. Este comportamiento permite el *polimorfismo de herencia*. Los programadores de C++ pueden emplear este comportamiento usando métodos virtuales [5].

2.3 Sobrecarga Dinámica: el Error

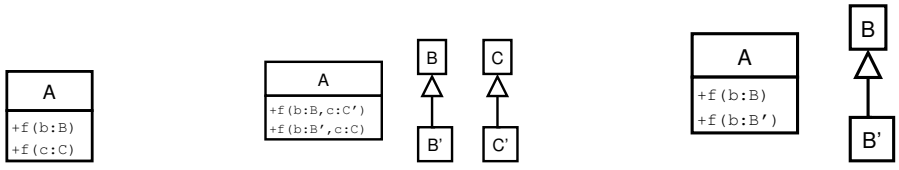
¿Cómo se determina si una regla es aplicable? En principio, tal y como se muestra en la figura 2(a), cada regla tiene un condicional que modela su aplicabilidad. Sin embargo, muchas veces la condición se determina únicamente por el tipo concreto del argumento del método. Por ejemplo, la regla `MoveRule` podría ser útil únicamente cuando las acciones son instancias de la clase `MoveAction`. Es este caso, como se muestra en la figura 2(b), el diseñador podría usar una aproximación mejor, evitando la comprobación explícita del tipo de la acción (i.e., *a* `instanceof MoveAction`).

Esta solución se puede interpretar de la siguiente manera: la clase `MoveRule` redefine el método `apply` *sólo* cuando el argumento es de tipo `MoveAction` (o algún subtipo). En caso contrario, se aplicaría el comportamiento por defecto heredado de la superclase. El problema aquí es que Java no considera esta versión del método `apply` una *sobreescritura parcial* del método de la superclase. En cambio, Java considera este método una versión *sobrecargada* del nombre `apply`.

En Java únicamente se permite la sobrecarga de métodos, al contrario que otros lenguajes de programación como C++, que permite, por ejemplo, la sobrecarga de operadores. La sobrecarga de métodos permite usar el mismo nombre para referirse a métodos *diferentes*. En principio, un conjunto de métodos sobrecargados no tienen ninguna relación semántica. Sin embargo, si el programador/diseñador da el mismo nombre a cosas diferentes, existe una alta probabilidad de que exista una relación entre estas cosas.

Java usa la firma del método (que debe ser única) para seleccionar el método a invocar. Cuando el compilador analiza la invocación de un método sobrecargado usa los tipos de los parámetros para seleccionar *estáticamente* (i.e., en tiempo de compilación) el método concreto que se invocará en tiempo de ejecución.

La figura 3(a) muestra un ejemplo: si $a \in A$ y $b \in B$, entonces $a.f(b)$ se interpreta estáticamente como una invocación de la versión de f con el parámetro B.



(a) (b) (c)

Figure 3: Ejemplos de sobrecarga

Esta idea simple se complica cuando se incluyen relaciones resultado de subtipados superclase-subclase o interface-implementación. Por ejemplo, como se muestra en la figura 3(b), si $a \in A$, $b \in B'$ y $c \in C'$, entonces

$a.f(b,c)$ podría ser una invocación de ambas versiones de f (el subtipado establece que $c \in C' \Rightarrow c \in C$ y $b \in B' \Rightarrow b \in B$). Consecuentemente, el compilador informa de un error de invocación ambigua de un método porque no puede tomar una decisión.

No obstante, algunos de estos conflictos se pueden resolver usando la estrategia de máxima especificidad. La figura 3(c) ilustra este escenario: si $a \in A$ y $b \in B'$, entonces $a.f(b)$ podría parecer otra invocación ambigua, pero no lo es. Ya que B' es más específico que B, el compilador seleccionará la segunda versión sin informar de ningún error.

Retomando el ejemplo de las reglas del juego (figura 2(b)), podemos observar que es muy similar al escenario del ejemplo anterior (figura 3(c)). La única diferencia entre ambas figuras redica en que las definiciones de los métodos se realizan en partes diferentes de la jerarquía de clases. Para todos los propósitos, la clase `MoveRule` tiene dos métodos `apply` sobrecargados. Tal y como ya hemos comentado, si en tiempo de compilación se infiere que $r \in$ `MoveRule` y $a \in$ `MoveAction` (figura 4(a)), entonces r.`apply`(a) invocará la versión más específica del método `apply`, sin ambigüedades.

Pero, ¿qué ocurre si, debido a la naturaleza estática del algoritmo, no es posible inferir que $a \in$ `MoveAction` en tiempo de compilación? La respuesta es que el compilador, estáticamente, invocará el método `apply` más general. Incluso si la instancia de `Action` asociada con A es de tipo `MoveAction` ($a \in$ `MoveAction`), el compilador invocará el método `apply` más general, ya que no puede inferir que $a \in$ `MoveAction` en tiempo de compilación. Sería necesario algún tipo de comprobación en tiempo de ejecución.

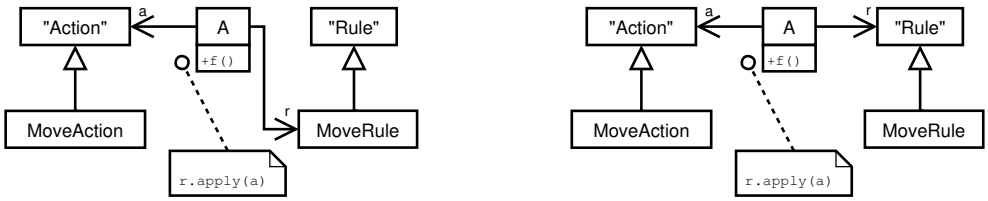


Figure 4: Ambigüedad en la invocación de métodos

Esta interpretación errónea de la operación interna del mecanismo de sobrecarga de Java se complica cuando el mecanismo de reescritura se confunde con el mecanismo de sobrecarga. La figura 4(b) muestra un ejemplo: se podría esperar que si $r \in$ `MoveRule` y $a \in$ `MoveAction`, entonces r.`apply`(a) invocaría el método de la subclase, y si $a \notin$ `MoveAction`, invocaría el método de la superclase.

3 Otro Escenario Problemático: Patrón Visitante

El problema presentado en la sección anterior está íntimamente ligado al que encotramos cuando empleamos la sobrecarga en Java para implementar el patrón de diseño Visitante [3].

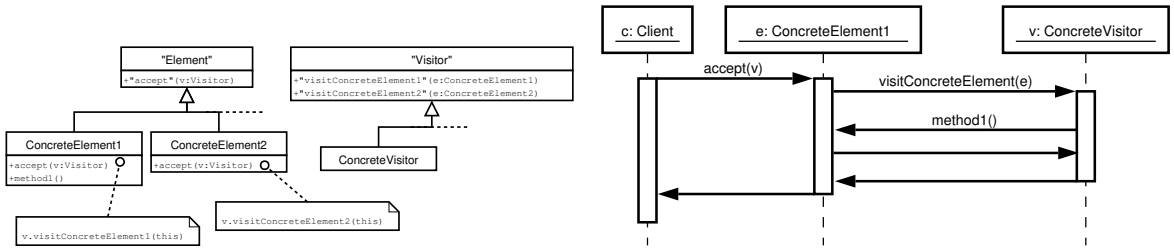
3.1 Patrón GoF Visitante

El patrón Visitante es una solución de la colección clásica GoF. Permite la definición de operaciones sobre una jerarquía de clases sin modificarla.

La razón principal para desligar algún comportamiento de las clases donde es útil, es evitar comprometer contínuamente la estabilidad de la jerarquía de clases cada vez que nuevos requerimientos imponen la creación de nuevas operaciones sobre la jerarquía. Aún más, esta separación de comportamiento permite cierta reusabilidad de las operaciones evitando arrastrar comportamientos innecesarios.

Tal y como ilustra la figura 5(a) las operaciones se separan de la jerarquía principal y se mueven a una nueva jerarquía de visitantes, donde cada clase tiene un método que, conceptualmente, está relacionado con el comportamiento de cada clase concreta de la jerarquía principal.

Por tanto, si un cliente tiene un elemento e (una instancia de cualquiera de las subclases de `Element`), entonces puede aplicar un visitante v sobre dicho elemento usando el método `accept`: e.`accept`(v). Cada clase concreta de elemento debe implementar la operación `accept`, seleccionando el método del visitante



(a) Visitante: diagrama de clases

(b) Visitante: diagrama de secuencia

Figure 5: Patrón de diseño GoF: Visitante

a invocar, y debe proveer la instancia del elemento necesaria para el visitante. La figura 5(b) muestra la secuencia habitual de pasos.

```
public abstract class Element {
  ...
  abstract public void accept(Visitor v);
  ...
}

public class ConcreteElement1 extends Element {
  ...
  public void accept(Visitor v) { v.visitConcreteElement1(this); }
  ...
}

public class ConcreteElement2 extends Element {
  ...
  public void accept(Visitor v) { v.visitConcreteElement2(this); }
  ...
}

public interface Visitor {
  public void visitConcreteElement1(ConcreteElement1 e);
  public void visitConcreteElement2(ConcreteElement2 e);
}

public class ConcreteVisitor implements Visitor {
  public void visitConcreteElement1(ConcreteElement1 e) { ... }
  public void visitConcreteElement2(ConcreteElement2 e) { ... }
}
```

Figure 6: Implementación en Java del Visitante

3.2 Implementación en Java

La implementación en Java del patrón de diseño Visitante es casi directa. Tal y como muestra la figura 6, es necesario definir un método abstracto `accept` en la raíz de jerarquía de clases `Element` e implementar este método en cada una de las clases concretas invocando el método de visitante pertinente. La jerarquía de clases `Visitante` tiene un interface como raíz donde se declarán todos los métodos de visita que deben implementar los visitantes concretos.

3.3 Limitaciones en la Factorización

Cada método visitante tiene un parámetro que representa la clase destino concreta. Entonces, como se muestra en la figura 7, se puede aplicar la sobrecarga de métodos para simplificar los nombres de los diferentes métodos de visita.

A continuación, este cambio implica cambiar también la jerarquía `Element`. Como ilustra la figura 8 la implementación del método `accept` debe modificarse para adecuarse al cambio.

```
public interface Visitor {
  public void visit(ConcreteElement1 e);
  public void visit(ConcreteElement2 e);
}

public class ConcreteVisitor implements Visitor {
  public void visit(ConcreteElement1 e) { ... }
  public void visit(ConcreteElement2 e) { ... }
}
```

Figure 7: Simplificación de los nombres de los métodos

```
public class ConcreteElement1 extends Element {
  ...
  public void accept(Visitor v) { v.visit(this); }
  ...
}

public class ConcreteElement2 extends Element {
  ...
  public void accept(Visitor v) { v.visit(this); }
  ...
}
```

Figure 8: Consideración de la sobrecarga

Nótese que el compilador (i.e., en tiempo de compilación) sabe que $this \in ConcreteElement1$ o $this \in ConcreteElement2$, y por lo tanto puede, estáticamente, seleccionar el método `visit` correcto sin ambigüedades.

En este momento, un vistazo rápido al código fuente de la figura 8 podría dar lugar a una interpretación errónea: cada subclase de `Element` implementa el método `accept` *de la misma manera* (v.`visit(this)`), por lo tanto, este comportamiento común puede factorizarse a la superclase `Element`. La figura 9 muestra el cambio.

Sin embargo, la factorización presentada en la figura 9 es errónea puesto que cada subclase `Element` implementa el método `accept` *de forma diferente*. Cada implementación del método `accept` se refiere a un método visitante distinto. En la figura 8, el compilador interpreta `this` con un tipo diferente para cada método `accept` (`ConcreteElement1` o `ConcreteElement2`), y puede resolver la sobrecarga. En el otro caso, después de factorizar el método `accept` a la superclase `Element` (figura 9), en tiempo de compilación `this` sólo se puede interpretar de tipo `Element` genérico. La selección del método visitante específico se debería realizar en función del tipo concreto del elemento (en tiempo de ejecución), pero esta aproximación es incompatible con la naturaleza estática del mecanismo de resolución de sobrecarga de Java.

4 Una Solución al Problema

4.1 Simular Sobrecarga Dinámica

Todos los problemas presentados en la sección 2 tiene como origen la naturaleza estática del proceso de resolución de la sobrecarga de métodos.

Combinando las técnicas de sobrecarga y reescritura es posible implementar un algoritmo dinámico

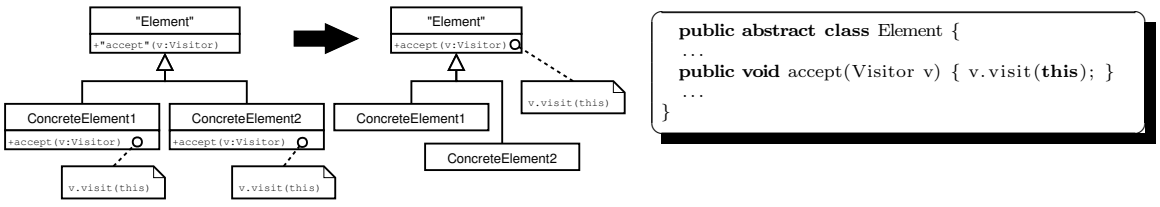


Figure 9: Factorizando el método `accept`

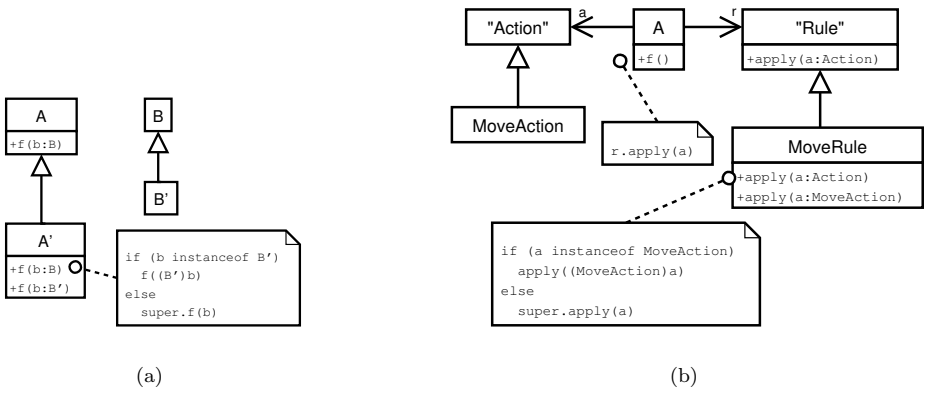


Figure 10: Simulación de sobrecarga dinámica

que invoque el método esperado incluso si los tipos concretos de los parámetros no se conocen en tiempo de compilación. Como muestra la figura 10, este comportamiento se puede lograr implementando tanto el método sobrecargado como un nuevo método que sobreescriba el método general y que realizará la invocación dinámica cuando sea necesario.

En el patrón de diseño Visitante no existe un método general para aplicar. Hay un conjunto de métodos en cada clase concreta. Por tanto, la factorización del método `accept` (figura 9) implica la creación de un método `visit` genérico en la superclase de la jerarquía de visitantes, el cual tendrá la responsabilidad de invocar el método concreto `visit` correcto en tiempo de ejecución (basándose en comprobaciones y conversiones de tipos). Como muestra la figura 11, como consecuencia de esta nueva funcionalidad, `Visitor` ya no puede ser una interface. Debe ser una clase abstracta con el Método Plantilla [3], cuya implementación está basada por completo en las operaciones `visit` originales. Más aún, sería una buena práctica la definición de la operación `visit` genérica como un método `final`, evitando que cualquier clase pudiese redefinir su comportamiento. En adición a esta pequeña mejora, sería una buena idea restringir la visibilidad de los métodos `visit` concretos. De esta forma, todas las invocaciones de los clientes (jerarquía de clases `Element`) se realizarían a través de la operación `visit` genérica.

```
public abstract class Visitor {

  public final void visit(Element e) {
    if (e instanceof ConcreteElement1)
      visit((ConcreteElement1) e);
    else if (e instanceof ConcreteElement2)
           visit((ConcreteElement2) e);
  }

  protected abstract void visit(ConcreteElement1 e);
  protected abstract void visit(ConcreteElement2 e);
}
```

Figure 11: Visitante con sobrecarga dinámica simulada

4.2 Alternativa Basada en Introspección

Los mecanismos de introspección del lenguaje Java suponen una aproximación mejor para evitar las limitaciones impuestas por la naturaleza estática de la sobrecarga de métodos. La idea principal es la siguiente: cuando se invoca un método sobrecargado, mediante los mecanismos de introspección exploramos el objeto destino, buscando el método apropiado en las *subclases concretas*. Como se especifica en [4] (sección 15.12.4.4, "Locate Method to Invoke"), si el método es un método de instancia y se invoca usando la búsqueda dinámica de métodos, se producirá la redefinición basada en el tipo en tiempo de ejecución del objeto destino.

El API de reflexión suministrado con Java, y usado en este escenario, es el siguiente:

- *Object* class. Devuelve el objeto `Class` que representa la clase del objeto para su futura introspección.

```
Class getClass()
```

- *Class* class. Obtiene un objeto respresentando un método con nombre `name` y un conjunto de parámetros que concide con los especificados en el array `parameters`. Si hay más de un métodos, se devuelve el más específico.

```
java.lang.reflect.Method getDeclaredMethod(String name, Class[] parameters)
                        throws NoSuchMethodException,
                               SecurityException
```

- *java.lang.reflect.Method* class. Invoca el método sobre el objeto `obj` usando `args` como los parámetros concretos.

```
Object invoke(Object obj, Object[] args)
        throws IllegalAccessException,
               IllegalArgumentException,
               InvocationTargetException
```

Por tanto, usando la interface de reflexión en el escenario que se muestra en el diagrama de clases de la figura 12, si $a \in A'$, y $b \in B$ o $b \in B'$, es posible invocar el método sobrecargado más específico dependiendo del tipo de b. La figura 12 ilustra una forma abreviada y otra paso a paso.

- Línea 4, obtener un objeto de clase a (objeto que representa la clase A').
- Línea 5, construir un array de objetos de tipo `Class` con las clases concretas de los parámetros de la invocación. En nuestro ejemplo, un único elemento que representa la clase concreta de b.
- Línea 6, introspección de la clase a buscando un método llamado "f" con la firma más específica, considerando los argumentos actuales.
- Líneas 7 a 8, invocar el método seleccionado.

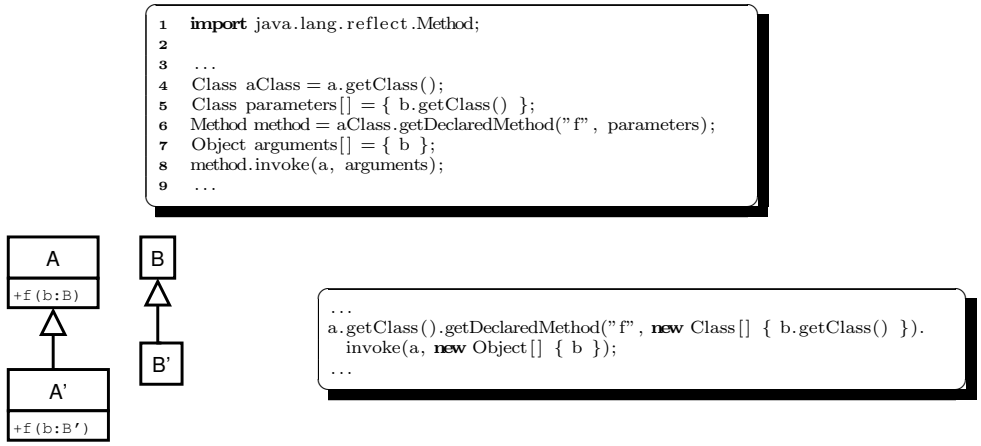


Figure 12: Ejemplo del API de reflexión de Java

Durante la invocación dinámica de un método algunas cosas pueden fallar. Si no existe el método adecuado (i.e., no hay ningún método con la firma requerida) se lanzará una excepción. Por eso, las invocaciones dinámicas deben encerrarse en un bloque try/catch o propagar las excepciones a un nivel superior.

Como se muestra en la figura 13, todo este comportamiento se puede mover a una clase auxiliar. Como consecuencia, el ejemplo de la figura 12 se reescribiría como:

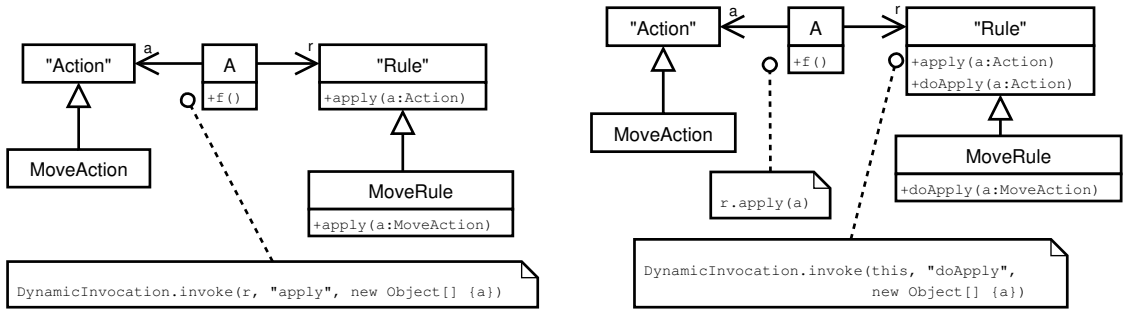
```
try {
  DynamicInvocation.invoke(a, "f", new Object[] { b });
} catch (...) { ... }
```

```
public class DynamicInvocation {
  public static Object invoke(Object obj, String name, Object[] args)
  throws java.lang.NoSuchMethodException,
         java.lang.reflect.InvocationTargetException {
    Class params[] = new Class[args.length];
    for (int i=0; i<args.length; i++)
      params[i] = args[i].getClass();
    try {
      return obj.getClass().getDeclaredMethod(name, params).invoke(obj, args);
    } catch (java.lang.reflect.InvocationTargetException e) {
      throw(e); // Method throws exception
    } catch (NoSuchMethodException e) {
      throw(e); // Name+signature method not found
    } catch (Exception e) {
      e.printStackTrace();
      throw(new RuntimeException(e.getMessage()));
    }
  }
}
```

Figure 13: Clase auxiliar para la invocación dinámica

La figura 14(a) muestra cómo se puede usar esta herramienta nueva para resolver el problema inicial presentado en este artículo.

Esta aproximación tiene una desventaja importante: los clientes deben ser conscientes de la necesidad de invocar algunos métodos de forma diferente, puesto que están usando sobrecarga dinámica. Esto puede mejorarse moviendo esta responsabilidad a la superclase `Rule`, añadiendo un método plantilla `apply` genérico, que se encargue de realizar las invocaciones dinámicas de los métodos sobrecargados. Esta mejora de la solución se muestra en la figura 14(b). Como se puede ver, el cliente (A en el ejemplo) no conoce nada acerca de las características especiales del método `apply`.



(a) Aplicación de la invocación dinámica

(b) Cambiando las responsabilidades de las clases

Figure 14: Invocación dinámica

La última mejora implica que la operación original tiene que dividirse en un método general y un conjunto de métodos auxiliares. Se puede evitar este inconveniente creando una invocación dinámica en la que *se invoca*

siempre la versión más específica del método. Por tanto, es necesario comprobar si el método seleccionado tiene una firma con un parámetro diferente (i.e., parámetro de tipo más específico) que el método de la clase original. Para dar soporte a este comportamiento emplearemos una implementación sobrecargada alternativa al método de invocación dinámica. La figura 15 muestra dicha implementación. Como se puede observar, existe un parámetro adicional: la clase base desde la cual se define el mecanismo de sobrecarga dinámica.

```
import java.lang.reflect.Method;

public class DynamicInvocation {
  public static Object invoke(...) throws ... { ... }

  public static Object invoke(Object obj, String name, Object[] args, Class root)
  throws java.lang.NoSuchMethodException,
         java.lang.reflect.InvocationTargetException {
    Class params[] = new Class[args.length];
    for (int i=0; i<args.length; i++)
      params[i] = args[i].getClass();
    try {
      Method method = obj.getClass().getDeclaredMethod(name, params);
      Method rootMethod = root.getDeclaredMethod(name, params);
      if (method.equals(rootMethod))
        throw(new NoSuchMethodException("Specific " + name + " version not found"));
      else
        return method.invoke(obj, args);
    } catch (java.lang.reflect.InvocationTargetException e) {
      throw(e); // Method throws exception
    } catch (NoSuchMethodException e) {
      throw(e); // Name+signature method not found
    } catch (Exception e) {
      e.printStackTrace();
      throw(new RuntimeException(e.getMessage()));
    }
  }
}
```

Figure 15: Mejora de la invocación dinámica

Finalmente, el nuevo mecanismo de invocación dinámica se puede emplear para eliminar los inconvenientes de la solución presentada en la figura 14(b). Ahora, tal y como se muestra en la figura 16, es suficiente un método `apply`. Este método puede sobreescribirse o sobrecargarse parcialmente sin ninguna consideración adicional en las subclases o los clientes. Se debe prestar atención para usar la versión de cuatro argumentos del método `invoke`. En otro caso, la versión de tres parámetros siempre invocará el método genérico a menos que existiese una versión más específica.

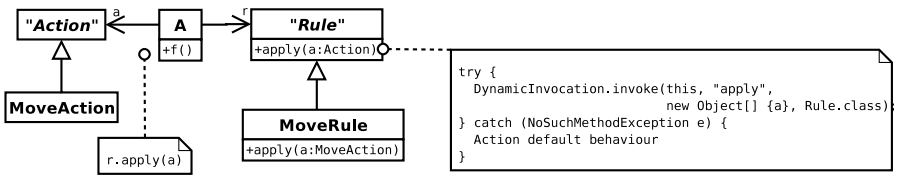


Figure 16: Eliminando métodos auxiliares

Finalmente, como se muestra en la figura 17, la misma solución se puede aplicar al escenario del patrón de diseño Visitante. Se necesitará un método genérico `visit` adicional en la superclase `Visitor`, el cual realizará la invocación dinámica dependiendo del tipo concreto. En esta nueva situación, el método `accept` se puede factorizar en la superclase `Element`.

5 Un Problema sin Solución

En este momento, podríamos pensar que hemos solucionado los problemas causados por la naturaleza estática del proceso de resolución de la sobrecarga de métodos en Java. El arreglo basado en introspección es una solución general y elegante, sin embargo, nos encontramos con otro escenario problemático con una solución más compleja.

```
public abstract class Visitor {

  public final void visit(Element e) {
    try {
      DynamicInvocation.invoke(this, "visit",
        new Object[] { r }, Visitor.class)
    } catch (Exception e) {...}
  }

  abstract public void visit(ConcreteElement1 e);
  abstract public void visit(ConcreteElement2 e);
}
```

```
public abstract class Element {
  ...
  public void accept(Visitor v) { v.visit(this); }
  ...
}
```

Figure 17: Visitante con invocación dinámica

En este caso el escenario aparece en un sistema de información para la gestión de riesgos desarrollado parcialmente usando el lenguaje de programación Java (ARMISTICE, Advanced Risk Management Information System: Tracking Insurances, Claims and Exposures [2]). Esta aplicación tiene una interfaz de usuario compleja, basada en Swing, con muchas ventanas de diálogo modales y no-modales. Todas estas ventanas de diálogo deben respetar su jerarquía dentro de la interfaz de usuario, recordando su ventana padre (otro diálogo o frame).

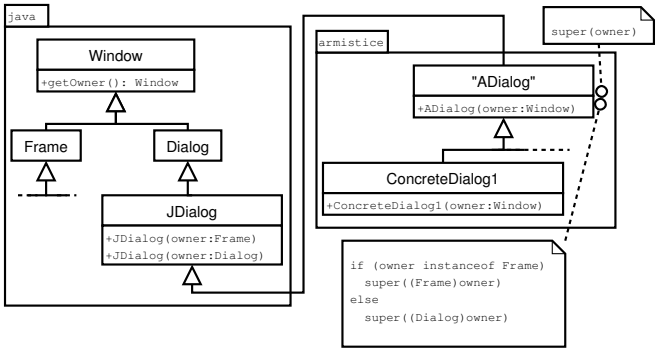


Figure 18: Jerarquía de diálogos en ARMISTICE

La figura 18 muestra una visión simplificada de la jerarquía de diálogos en la aplicación. En primer lugar, cada ventana en una interfaz de usuario desarrollado con Swing tiene como clase raíz `Window`. Esta clase no se extiende directamente por las clases de la aplicación. En su lugar, existen una serie de clases intermedias como `Frame`, `JFrame`, `Dialog` y `JDialog`. En nuestra aplicación todos los diálogos concretos extienden la clase `ADialog`.

Cada diálogo del sistema tiene que conocer su ventana padre. Por tanto, el constructor de cada clase concreta de diálogo recibe el padre como parámetro `owner` (tipo `Window`). Sin embargo, la superclase `JDialog` tiene un constructor con un parámetro `owner` de tipo `Frame`, y otro con un parámetro `owner` de tipo `Dialog`. De nuevo, no es posible seleccionar el constructor de forma estática y el compilador informa de un error de invocación ambigua a un método (constructor). Como primera aproximación, la figura 18 muestra un intento de aplicar la solución propuesta en la sección 4.1. En cualquier caso, esta técnica no se puede aplicar aquí puesto que el compilador obliga a que la instrucción `super` sea la primera en la implementación del constructor (esto asegura que el estado del objeto se inicializa antes de cualquier operación; un compilador más avanzado podría permitir nuestras conversiones de tipo). Por otra parte la técnica basada en introspección no tiene sentido en este caso.

Por tanto, no es posible implementar el comportamiento deseado tal y como se muestra en la figura 18. La única solución es añadir dos constructores en toda la jerarquía `ADialog`: uno con un `Frame` como padre y otro con `Dialog`.

5.1 Más Introspección

Continuando en el escenario anterior, asumiendo $w \in$ `Window`, ¿cómo se puede crear un objeto `ConcreteDialog1` que tenga w como su padre? En este caso es fácil aplicar la solución propuesta en la sección 4.1. Sin embargo, la técnica basada en introspección presentada en la sección 4.2 no se puede aplicar directamente, pero se puede extender para dar soporte a las invocaciones dinámicas de constructores.

```
public class DynamicInvocation {

  ...

  public static Object newInstance(Class cls, Object[] args)
  throws java.lang.NoSuchMethodException, java.lang.InstantiationException,
         java.lang.reflect.InvocationTargetException {
    Class params[] = new Class[args.length];
    for (int i=0; i<args.length; i++)
      params[i] = args[i].getClass();
    try {
      return cls.getConstructor(params).newInstance(args);
    } catch (java.lang.reflect.InvocationTargetException e) {
      throw(e); // Method throws exception
    } catch (NoSuchMethodException e) {
      throw(e); // Name+signature method not found
    } catch (Exception e) {
      throw(new RuntimeException(e.getMessage()));
    }
  }
}
```

Figure 19: Creación dinámica de objetos

La figura 19 muestra una clase auxiliar implementada usando una vez más el API de reflexión. Se debe usar como sigue:

```
try {
  DynamicInvocation.newInstance(ConcreteDialog1.class, new Object[] { w });
} catch (...) { ... }
```

6 Conclusiones

En este artículo hemos presentado algunos de los problemas causados por soporte estático de la sobrecarga de métodos en el lenguaje de programación Java. La cuestión ha sido explorada desde la invocación de métodos hasta la instanciación de objetos, y desde el punto de vista del patrón de diseño Visitante hasta la perspectiva de una aplicación Java del mundo real.

Hemos propuesto y presentado en detalle una solución a nivel de aplicación, explicando sus consecuencias, su ventajas y desventajas. También hemos identificado un escenario problemático, donde no se ha podido encontrar una solución satisfactoria a nivel de aplicación.

References

[1] Java 2 SDK, Standard Edition Documentation. `http://java.sun.com/j2se`.

[2] David Cabrero, Carlos Abalde, Carlos Varela, and Laura Castro. ARMISTICE: an experience developing management software with Erlang. In *Proceedings of the 2003 ACM SIGPLAN workshop on Erlang*, pages 23–28. ACM Press, 2003.

[3] Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns Elements of Reusable Object-Oriented Software*. Addison-Wesley, Massachusetts, 1995.

[4] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification Second Edition*. Addison-Wesley, Boston, Mass., 2000.

[5] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Longman Publishing, 2000.

Memory consumption analysis of Java smart cards

Pablo Giambiagi
Swedish Institute of Computer Science, Box 1263, SE-164 28, Kista, Sweden.
E-mail: {pablo at sics.se}

and

Gerardo Schneider*
Dept. of Informatics, University of Oslo - PO Box 1080 Blindern, N-0316 Oslo, Norway
E-mail: {gerardo at ifi.uio.no}

Abstract

Memory is a scarce resource in Java smart cards. Developers and card suppliers alike would want to make sure, at compile- or load-time, that a Java Card applet will not overflow memory when performing dynamic class instantiations. Although there are good solutions to the general problem, the challenge is still out to produce a static analyser that is certified and could execute on-card. We provide a constraint-based algorithm which determines potential loops and (mutually) recursive methods. The algorithm operates on the bytecode of an applet and is written as a set of rules associating one or more constraints to each bytecode instruction. The rules are designed so that a certified analyser could be extracted from their proof of correctness. By keeping a clear separation between the rules dealing with the inter- and intra-procedural aspects of the analysis we are able to reduce the space-complexity of a previous algorithm.

1 Introduction

Embedded systems, special-purpose computer systems built into larger devices, can be found almost everywhere: from a simple coffee machine, to a mobile phone and a car, all may contain at least one embedded system, if not many. Programmable smart cards are small personal devices provided with a microprocessor capable of manipulating confidential data, allowing the owner of the card to have secure access to chosen applications. Applications, called applets, can be downloaded and executed in these small communicating devices, raising major security issues. Indeed, without appropriate security measures, a malicious or simply buggy applet could be installed in smart cards and compromise sensitive data, perform unauthorised transactions or even render the card useless by consuming all of its resources.

The multi-application model, i.e. the ability to load applets from possibly competing providers, puts strong demands on the reliability of the software used to manipulate the data entrusted to the card. Hence, it is essential that the platform, as well as the applets running on it, satisfy a minimum of safety and security guarantees in order to preserve confidentiality, integrity and availability of information. To guarantee availability of the services offered by small devices the management and control of resources (e.g. memory) is crucial.

These days a top-notch smart card has about 64KB EEPROM, 200KB ROM, 4KB RAM and a 32-bits data bus. The corresponding numbers for banking and other low-end cards are 16KB, 16KB, 1KB and 8 bits. In the smart card world, memory is a very precious resource which must be manipulated carefully. Consequently the smart card industry has developed specific programming guidelines and procedures to keep memory usage under control. We quote Chen: "Because memory in a smart card is very scarce, neither persistent nor transient objects should be created willy-nilly" [7, Section 4.4]. The advice is extended to method invocations: "You should also limit nested method invocations, which could easily lead to stack overflow. In

*Partially supported by the *RNTL* French project, CASTLES (*Conception d'Analyses Statiques et de Tests pour le Logiciel Embarqué Sécurisé*). Part of this work was done while the author was spending one year as a researcher at IRISA-INRIA.

particular, applets should not use recursive calls" [7, Section 13.3]; and object allocation: "An applet should always check that an object is created only once" [7, Section 13.4]. Even though these recommendations are generally respected by the industry, mistakes regarding memory usage –like the instantiation of a class inside a loop– either accidental or intentional, may have dire consequences. In fact, nothing prevents a(n) (intentionally) badly written applet to allocate all persistent memory on a card. Hence, the ability to detect recursive methods and loops is imperative, both during the development process and at applet load-time.

As far as we know there is no *on-card* tool available for detecting the dynamic instantiation of classes inside cycles and/or recursive functions for Java smart cards. Leroy [10, 11] describes a bytecode verifier that could be made to run on-card, but it does not address properties related to memory usage. Previous work presents a certified analyser to determine loops and mutually recursive methods but its memory footprint prevents it from being deployed on-card [6].

This paper takes up the challenge of constructing a formally certified static analyser whose memory requirements are low enough that it could be added, eventually, to an on-card bytecode verifier. This implies making the right trade-off between the precision of the analysis and the practical viability of the formal certification process. Regarding the latter requirement we have adopted the approach introduced in [6]. According to this approach, the analyser should be first described as a constraint-based algorithm, and then formally extracted from the proof of correctness of the algorithm in the proof assistant Coq [4]. Whereas [6] demonstrates the feasibility of the extraction process, the difficulty remains on how to describe more efficient algorithms (in terms of memory consumption) without compromising the ability to perform code extraction.

The algorithms presented here improve those presented in [6], in terms of the auxiliary memory used and because they also cover subroutines and exceptions, which were not addressed originally. Although the memory consumption is not fully optimised, we have taken care to partition the tasks to reduce the amount of data that needs to be kept simultaneously in memory.

One further feature of our algorithm is that it works directly on the bytecode and there is no need to precede its execution with the construction of extra data structures (e.g., a control-flow graph). Our approach includes both intra- and inter-procedural analyses. Both work on arbitrary bytecode, i.e. it is not assumed that the bytecode is produced by a "well-behaved" compiler. For the intra-procedural analysis, the construction of the graph of basic blocks –a prerequisite of many textbook algorithms– is done implicitly without resorting to any auxiliary, pre-existent analysis.

The language being considered is the *bytecode* language JCVML (Java Card Virtual Machine Language), although the techniques discussed in this paper are independent of this choice and can be applied to most bytecode languages. JCVML manipulates (dynamic) objects and arrays and besides the usual stack and register operations it accommodates interesting features like virtual method calls, (mutually) recursive methods, (un)conditional jumps, exceptions and subroutines. We assume there is no garbage collector, which is the case for Java Card up to version 2.1[1].

The paper is organised as follows. Section 2 briefly presents the language being considered while in Section 3 we present the specification of the algorithm. We prove, in Section 4, termination of our algorithm as well as its soundness and completeness with respect to an abstraction of the operational semantics of the language. In Section 5 we show how we could treat subroutine calls and exceptions. In the last section we discuss the complexity of our algorithm, related and future work.

2 The language

We consider in this paper the whole JCVML language. The instruction set, as well as an operational semantics of a language that models JCVML is given in [16]. It comprises, among others, the following instruction sets:

- Stack manipulation: `push`, `pop`, `dup`, `dup2`, `swap`, `numop`;
- Local variables manipulation: `load`, `store`;
- Jump instructions: `if`, `goto`;
- Heap manipulation: `new`, `putfield`, `getfield`;

[1]Starting with Java Card 2.2 the machine includes a garbage collector which may be activated invoking an API function at the end of the execution of the applet.

- Array instructions: `arraystore`, `arrayload`;
- Method calls and return: `invokevirtual`, `invokestatic`, `invokespecial`, `invokeinterface`, `return`.

In Section 5 we sketch how to extend the analysis to cover subroutine calls and exception handling.

A JCVML program P (which will be called in what follows a *bytecode program* or simply, a *program*) is a set of methods $m \in \mathit{Method}_P$ consisting of a numbered sequence of instructions. We write $\mathit{InstrAt}(m, pc) = \mathtt{i}$ to denote that the instruction at program line pc (usually called a *pc-number*) in method m is `i`. Let $\mathit{ProgCounter}_P$ be the set of all the pc-numbers. We will usually omit the subscript P, being understood that the analysis depends on a given program P.

3 Specification of the analysis

We present in this section a constraint-based specification of an analyser for detecting the occurrence of a `new` instruction inside potential cycles, due to intra-method loops and/or (mutually) recursive method calls. It consists in the computation of three functions *Loop*, *Loop*' and *Rec* respectively providing information about intra-method cycles, methods called from intra-method cycles and (mutually) recursive methods (as well as the methods reachable from these). Using the above functions, the main algorithm checks whether a `new` instruction occurs inside a potential cycle. The specification of the main algorithm and all the above-mentioned functions are presented as a set of rules, each rule associating one or more constraints to each instruction in the program. The solution of the set of constraints (which are of the form $\{F_1(X) \sqsubseteq X, \ldots F_n(X) \sqsubseteq X\}$) is the least fix-point of a function $\mathcal{F}$ obtained from $F_1, \ldots F_n$. By a corollary to Tarski's Fix-point Theorem, this element may be obtained as the limit of the stabilising sequence $(\mathcal{F}^n(\bot))_{n\in\mathbb{N}}$.

We show next how to detect intra-method loops and (mutually) recursive methods.

3.1 Detection of loops

We will define two functions, *Loop* and *Loop*', for detecting cycles in a given program P: *Loop* will be defined intra-procedurally while *Loop*' will propagate inter-procedurally the results given by *Loop*.

Intra-procedural analysis. The analysis works by identifying the basic blocks in each method, and by associating to each program point the set of basic blocks that may be traversed on an execution path ending in the point in question. With this information at hand, the analysis is able to mark the instructions involved in a potential cycle. The concept of *basic block* is well-established in program analysis [14]. It refers to a contiguous, single-entry code segment whose (conditional) branching instructions may only appear at the end of the block. This implies that, with the exception of the very first block (which starts at $pc = 1$), all other basic blocks start at the destination of some jump. Our analysis identifies each basic block by the pc-number of its first instruction, which is found observing the targets of branching instructions.

We proceed to formalise this intuition. Given a program P, let *Method* and *ProgCounter* be respectively the sets of method names and pc-numbers of P. Given a method m, its pc-numbers range from 1 to the constant END_m, which is equal to the number of lines of method m plus one. The analysis defines the function

$$\mathit{Loop} \colon \mathit{Method} \times \mathit{ProgCounter} \to \wp(\mathit{ProgCounter} \uplus \{\bullet\}),$$

where $\mathit{Loop}(m, pc) \cap \mathit{ProgCounter}$ is the set of basic blocks (identified by their first location) that may be traversed on an execution path to (m, pc). Moreover, if $\bullet \in \mathit{Loop}(m, pc)$, then we know that the location (m, pc) lies in a cycle of the control flow graph of P.

We do not assume any structure on the bytecode program being considered, except that each `goto` is intra-method – a property guaranteed by the bytecode verifier [10]. Table 1 shows the rules (one for each instruction) used for computing *Loop*. That is, *Loop* is defined as the least element of the lattice

$$\mathcal{L} = (\mathit{Method} \times \mathit{ProgCounter} \to \wp(\mathit{ProgCounter} \uplus \{\bullet\}),\ \sqsubseteq),$$

that satisfies all the constraints derived from the code using the rules in Table 1. The order relation of the lattice is defined as $f \sqsubseteq f'$ iff $f(m, pc) \sqsubseteq f'(m, pc)$, for all (m, pc). Notice that the co-domain of *Loop* is a powerset. This is a lattice under subset inclusion; its least element, $\bot$, is the empty set.

$$\frac{}{\{1\} \sqsubseteq Loop(m,1)} \quad (1)$$

$$\frac{(m,pc) : \texttt{goto}\ pc'}{F(Loop(m,pc),pc') \sqsubseteq Loop(m,pc')} \quad (2)$$

$$\frac{(m,pc) : \texttt{if}\ t\ op\ \texttt{goto}\ pc'}{\begin{array}{l} F(Loop(m,pc),pc') \sqsubseteq Loop(m,pc') \\ F(Loop(m,pc),pc+1) \sqsubseteq Loop(m,pc+1) \end{array}} \quad (3)$$

$$\frac{(m,pc) : \texttt{invokevirtual}\ m'}{Loop(m,pc) \sqsubseteq Loop(m,pc+1)} \quad (4)$$

$$\frac{(m,pc) : \texttt{return}}{\bot \sqsubseteq Loop(m,\mathrm{END}_m)} \quad (5)$$

$$\frac{(m,pc) : \texttt{instr}}{Loop(m,pc) \sqsubseteq Loop(m,pc+1)} \quad (6)$$

Table 1: Rules for *Loop*

Rule (1) labels the first line of each method as a basic block. Although this rule is not needed for detecting loops, it helps to identify which loops are actually reachable (see Example 1). Rule (2) serves two simultaneous purposes: It records location (m, pc') as the start of a basic block, and takes care of detecting cycles. If the list of blocks possibly visited up to (m, pc) contains the destination of the `goto` instruction, i.e. $pc' \in Loop(m, pc)$, the rule marks (m, pc') as belonging to a cycle. These two tasks are achieved with the help of the following auxiliary function:

$$F(L_{m,pc}, pc') = \begin{cases} L_{m,pc} \cup \{\bullet\} & \text{if } pc' \in L_{m,pc} \\ L_{m,pc} \setminus \{\bullet\} \cup \{pc'\} & \text{otherwise} \end{cases} \quad (7)$$

where "\" is the usual set subtraction operator.

A conditional branch instruction determines the starting point of two basic blocks: (1) the destination of the jump and (2) the location of the instruction immediately after the conditional jump. This is taken care of by rule (3) again using function F.

Rule (4) concerns virtual method invocations. As we are considering here only intra-procedural cycles, the content of *Loop* is not propagated to method m'. Similar rules for `invokestatic`, `invokespecial` and `invokeinterface` may be considered. The `return` instruction –rule (5)– does not propagate anything, as expected for an intra-procedural analysis. In rule (6), `instr` stands for any other instruction not defined by rules (1)–(5); in this case the information is simply propagated to the next instruction.

For the sake of simplicity define a predicate $Loop_{m,pc} \equiv (\{1, \bullet\} \subseteq Loop(m, pc))$, so that $InstrAt(m, pc)$ is in a reachable loop iff $Loop_{m,pc}$.

Example 1 *In Fig. 1 we show part of three bytecode methods; the value of Loop appears to the right of each program. We assume that the lines marked with ellipsis have no branching instructions. Program (a) contains no cycles, which is reflected by the absence of $\bullet$ in the right column. Program (b) has a cycle involving lines 20 through 70. Notice how the analysis discovers the basic blocks (with start in lines 1, 20, 31, 41, 50, 51 and 90). Observe as well that $\bullet$ is not propagated to line 90 by the branching instructions at lines 40 and 50. The annotation is filtered by F (in rule 3) to reflect the fact that line 90 lies outside the cycle. Finally, Program (c) illustrates the case of an unreachable cycle. Lines 2-3 are marked as belonging to a cycle involving the basic block starting in line 2, but there is no path to this block from line 1.* □

Inter-procedural analysis. Given a program P, we define the following lattice:

$$\mathcal{L} = (Method \times ProgCounter \to \{\bot, \top\},\ \sqsubseteq),$$

where *Method* and *ProgCounter* are as before and $\{\bot, \top\}$ is the usual lattice with $\bot \sqsubseteq \top$. The order for the function lattice is defined as follows: $f \sqsubseteq f'$ if and only if $f(m, pc) \sqsubseteq f'(m, pc)$, for all (m, pc). For convenience and to be consistent with the notation used in the computation of *Loop*, we will write $\bullet$ for $\top$.

Loop$'$ captures all the instructions reachable from intra-procedural cycles through method calls. It is defined by the rules shown in Table 2. To keep the presentation simple the first constraint for rule (8) has been written $\bullet \sqsubseteq Loop'(m', 1)$, but it must be understood as: $\forall m_{ID} \in implements(m'), \bullet \sqsubseteq Loop'(m_{ID}, 1)$, and similarly for rule (9). The same remark holds for the `invokevirtual` rules of the function *Rec* defined later. Function *implements* is a static over-approximation of the dynamic `methodLookup` function [16], which

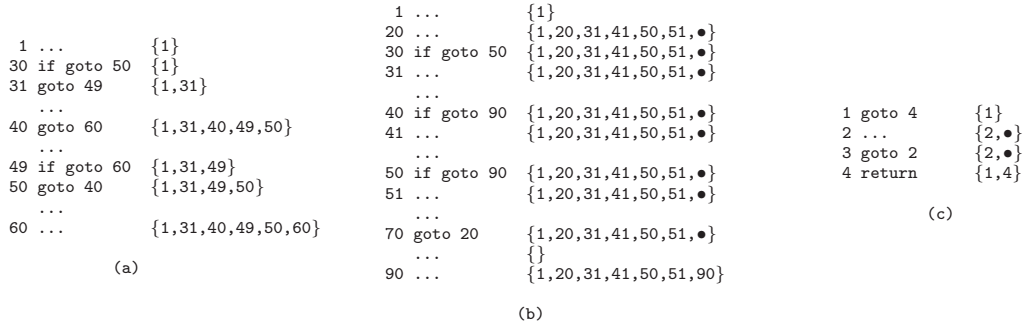


Figure 1: Example

$$\frac{(m,pc) : \texttt{invokevirtual}\, m' \quad Loop_{m,pc}}{\begin{array}{l}\bullet \sqsubseteq Loop'(m',1)\\ Loop'(m,pc) \sqsubseteq Loop'(m,pc+1)\end{array}} \quad (8)$$

$$\frac{(m,pc) : \texttt{invokevirtual}\, m' \quad \neg Loop_{m,pc}}{\begin{array}{l}Loop'(m,pc) \sqsubseteq Loop'(m',1)\\ Loop'(m,pc) \sqsubseteq Loop'(m,pc+1)\end{array}} \quad (9)$$

$$\frac{(m,pc) : \texttt{instr}}{Loop'(m,pc) \sqsubseteq Loop'(m,pc+1)} \quad (10)$$

$$\frac{(m,pc) : \texttt{return}}{\bot \sqsubseteq Loop'(m,\mathrm{END}_m)} \quad (11)$$

Table 2: Rules for $Loop'$

returns all possible implementations of a given method with name m' relative to a program P. Notice that on Java cards the information needed to compute such function is available at load-time. We do not specify it in further detail.

3.2 Detection of (mutually) recursive methods

Given a program P, we define a lattice $\mathcal{L}$ as follows:

$$\mathcal{L} = (Method \times ProgCounter \times \rightarrow \wp(Method \uplus \{\bullet\}),\ \sqsubseteq),$$

where the order relation is inherited from the powerset lattice in the usual way, i.e. $f \sqsubseteq f'$ iff $f(m,pc) \sqsubseteq f'(m,pc)$, for all (m,pc). *Rec* takes values over the above lattice. The definition of *Rec* is given by the rules described in Table 3. The first rule corresponds to the case of a recursive method m; it annotates the first instruction of the method with $\bullet$ and the method name. The application of the other rules will then propagate these annotations to all the instructions in the method. Rule (13) shows the case of a non self-invocation. The content of *Rec* is propagated unchanged to the next instruction inside the method, and to $InstrAt(m',1)$ as determined by function $G : \wp(Method \uplus \{\bullet\}) \times Method \longrightarrow \wp(Method \uplus \{\bullet\})$:

$$G(R_{m,pc}, m') = \begin{cases} R_{m,pc} \cup \{m, \bullet\} & \text{if } m' \in R_{m,pc} \\ R_{m,pc} \cup \{m\} & \text{if } m' \notin R_{m,pc} \end{cases}$$

At each method call $(m,pc) : \texttt{invokevirtual}\, m'$, G adds to $Rec(m,pc)$ the calling method name; if the called method name is already in $Rec(m,pc)$, then also $\bullet$ is added. Intuitively, G detects whether a given method has been visited more than once following the same "path". The other rules only propagate the result defined by the rules corresponding to `invokevirtual` (as before, `instr` stands for any instruction not already covered by rules (12)–(14)).

For a given method m and program counter pc, we define the predicate $Rec_{m,pc} \equiv (\bullet \in Rec(m,pc))$.

Remark. Notice that we are interested in knowing which methods may be executed an unknown number of times due to recursion. Thus, *Rec* detects not only all the (mutually) recursive methods but also all the methods which are called from those: for any method m such that $Rec_{m,1}$, if $m \in Rec(m,1)$, m is (mutually) recursive, otherwise m is reachable from a (mutually) recursive method. From now on we will say that a method is recursive if it calls itself or if it belongs to a set of mutually recursive methods.

$$\frac{(m,pc) : \texttt{invokevirtual}\, m' \quad m = m'}{\begin{array}{l} Rec(m,pc) \cup \{m, \bullet\} \sqsubseteq Rec(m',1) \\ Rec(m,pc) \sqsubseteq Rec(m,pc+1) \end{array}} \quad (12)$$

$$\frac{(m,pc) : \texttt{invokevirtual}\, m' \quad m \neq m'}{\begin{array}{l} G(Rec(m,pc), m') \sqsubseteq Rec(m',1) \\ Rec(m,pc) \sqsubseteq Rec(m,pc+1) \end{array}} \quad (13)$$

$$\frac{(m,pc) : \texttt{return}}{Rec(m,pc) \sqsubseteq Rec(m, \mathrm{END}_m)} \quad (14)$$

$$\frac{(m,pc) : \texttt{instr}}{Rec(m,pc) \sqsubseteq Rec(m,pc+1)} \quad (15)$$

Table 3: Rules for *Rec*

3.3 Main algorithm

In this section we present the specification of the main algorithm, which uses the three analyses described so far.

The only instructions sensitive to our analysis are the ones that enlarge the heap, namely instructions that create array objects and class instances (see the operational semantics in [16]). For simplicity we consider here only one instruction `new`, but we mean both $\texttt{new}(cl)$ and $\texttt{new}(\mathsf{array}\, a)$, where a is an array type. We write $Cycle_{m,pc}$ as a shortcut for $Loop_{m,pc} \vee Loop'_{m,pc} \vee Rec_{m,pc}$. The specification of the algorithm is given by the following three-valued function $\Gamma : Method \times ProgCounter \to \{0, 1, \infty\}$:

$$\Gamma(m,pc) = \begin{cases} \infty & \text{if } (m,pc) : \texttt{new}(cl) \ \wedge \ Cycle_{m,pc} \\ 1 & \text{if } (m,pc) : \texttt{new}(cl) \ \wedge \ \neg Cycle_{m,pc} \\ 0 & \text{otherwise} \end{cases}$$

So $\Gamma(m,pc)$ represents a bound on the number of object instances that may be created by the program instruction at (m,pc). If there is no `new` instruction there, then $\Gamma(m,pc)$ is clearly 0. When $(m,pc) : \texttt{new}(cl)$ and (m,pc) lies in no (potential) cycle, then we know that the instruction may be executed at most once and therefore $\Gamma(m,pc) = 1$. Finally, $\Gamma(m,pc) = \infty$ if the current instruction is a `new` and lies inside a potential cycle. Notice that the main algorithm is obtained without computing a fix-point, since all the information is already in *Cycle*.

4 Properties of the analysis

4.1 Termination

One of the crucial properties for proving termination of our analyser is the *ascending chain condition*, i.e. that the underlying lattices have no infinite, strictly increasing chains. The property is trivially satisfied by all our lattices as their height are finite. The algorithm reduces to the problem of solving a set of constraints over a lattice $\mathcal{L}$, which can be transformed into the computation of a fix-point of a monotone function over $\mathcal{L}$. Termination follows from the proof of the termination of the fix-point computation for obtaining *Loop*, *Loop*', *Rec* and Γ.

We need the following auxiliary lemma about the functions F and G used in the definition of *Loop* and *Rec*.

Lemma 1 *The functions F and G are monotone.* □

It is well known (see for instance [14]) that the solution of a set of constraints of the form $\{F_1(X) \sqsubseteq X, \ldots F_n(X) \sqsubseteq X\}$, where each F_i is monotone, is the least solution of a function $\mathcal{F}$ obtained from $F_1, \ldots F_n$. Moreover, by a corollary of Tarski's fix-point theorem, this element may be obtained as the limit of the stabilising sequence $(\mathcal{F}^n(\bot))_{n\in\mathbb{N}}$. As a corollary of the previous lemma we have the following result.

Corollary 1 *The computations of the least fix-points corresponding to the set of constraints defining Loop, Loop' and Rec always terminate.* □

Termination of the algorithm follows directly from termination of *Loop*, *Loop*' and *Rec*, given that the algorithm simply scan each method sequentially without iterating.

Theorem 1 *The computation of the function Γ always terminates.* □

4.2 Soundness and completeness

We consider here a *maximal semantics*. That is, the semantics of a program P, noted $[\![P]\!]$, is the set of all its possible executions (traces). Such traces may be obtained symbolically by applying the rules of the operational semantics [16]. We prove soundness and completeness of the functions introduced in Section 3 w.r.t. an appropriate abstraction of the operational semantics. For *Rec* we consider the usual method-call graph, which is an abstraction of the control-flow graph $\mathcal{G}(P)$. For the intra-procedural case (*Loop*) we consider $\mathcal{G}_m(P)$, which is a modified restriction of $\mathcal{G}(P)$ to the given method m. The trace obtained from a traversal of the graph $\mathcal{G}(P)$, which does not take into consideration the tests in branching instructions, is an *abstract trace* of P; let $[\![\widehat{P}]\!]$ denote the set of all the abstract traces of program P. In what follows we will use the fact that $[\![P]\!] \subseteq [\![\widehat{P}]\!]$ (see [13, 15] and references therein).

4.2.1 Loop

Given a program P, its *control-flow graph* $\mathcal{G}(P)$, is a 4-tuple $(\mathcal{S}, \rightarrow, s_0, F)$, where

- $\mathcal{S}$ is a set of *nodes* (or *program states*), ranging over elements of $Method \times ProgCounter$;
- $\rightarrow \subseteq \mathcal{S} \times \mathcal{S}$ is a set of *transitions*, which models the flow of control;
- s_0 is the initial state: $s_0 = (m_0, 1)$, where m_0 is the main method;
- $F = (m_0, \mathrm{END}_{m_0})$ is the final state.

In what follows we will write $\mathcal{G}$ instead of $\mathcal{G}(P)$. More formally, $\rightarrow$ is defined as follows:

$$\frac{(m,pc) : \texttt{goto}\ pc'}{(m,pc) \rightarrow (m,pc')} \qquad \frac{(m,pc) : \texttt{if}\ c\ \texttt{goto}\ pc'}{(m,pc) \rightarrow (m,pc') \qquad (m,pc) \rightarrow (m,pc+1)}$$

$$\frac{(m,pc) : \texttt{return}}{(m,pc) \rightarrow (m, \mathrm{END}_m)} \qquad \frac{(m,pc) : \texttt{invokevirtual}\ m'}{(m,pc) \rightarrow (m',1) \qquad (m', \mathrm{END}_{m'}) \rightarrow (m, pc+1)}$$

$$\frac{(m,pc) : \texttt{instr}}{(m,pc) \rightarrow (m,pc+1)}$$

where `instr` is any instruction different from `invokevirtual`, `goto`, `if`, and `return`. As usual, $\rightarrow^+$ denotes the transitive closure of $\rightarrow$; we say that a state s' is *reachable* from s iff $s \rightarrow^+ s'$ and write $s' \in Reach(s)$. We also introduce a *satisfaction* relation: $\mathcal{G} \models \phi$ iff $\mathcal{G}$ satisfies the property ϕ.

For our purposes it is convenient to define the control-flow graph of each method independently, which may be obtained from the graph $\mathcal{G}$ by defining a transition relationship $\rightarrow_m$ which agrees with $\rightarrow$ except for the `invokevirtual` instruction:

$$\frac{(m,pc) : \texttt{invokevirtual}\ m'}{(m,pc) \rightarrow_m (m,pc+1)}.$$

For each method m, we denote the graph obtained using the new transition $\rightarrow_m$ by $\mathcal{G}_m$, and call it the *m-control-flow graph*. We say that *pc*' is reachable from *pc* in $\mathcal{G}_m$, written $pc' \in Reach_m(pc)$, if and only if $(m,pc) \rightarrow_m^+ (m,pc')$. We will omit the subindex m and we will simply write *Reach* and $\rightarrow$ instead of $Reach_m$ and $\rightarrow_m$ respectively whenever understood from the context. We define the following predicate:

$$\mathcal{G}_m \models SynC(m,pc) \quad \text{iff} \quad \mathcal{G}_m \models pc \in Reach(pc).$$

Thus, the predicate *SynC* determines whether a given instruction is in a *syntactic cycle*. Notice that this predicate only characterises *intra-method* cycles.

The following theorem establishes that the function *Loop* characterises exactly all the syntactic cycles of a method.

Theorem 2 (Soundness and Completness of *Loop*) *$Loop_{m,pc}$ if and only if $\mathcal{G}_m \models SynC(m,pc)$.* □

4.2.2 Loop'

The following predicate, *SynCReach*, determines whether a given instruction is reachable from a method invocation inside a syntactic cycle:

$$\mathcal{G} \models \mathit{SynCReach}(m, pc)$$
$$\text{iff}$$
$$(\exists m')\ \mathcal{G}_{m'} \models \mathit{SynC}(m', pc') \text{ and } \mathcal{G} \models (m, pc) \in \mathit{Reach}(m', pc').$$

Loop' characterises exactly all instructions reachable from a cycle:

Theorem 3 (Soundness and Completeness of *Loop*') *$\mathit{Loop}'_{m,pc}$ if and only if $\mathcal{G} \models \mathit{SynCReach}(m, pc)$.* □

4.2.3 Rec

Given a program P, its *method-call graph* $\mathcal{M}(P)$, is a 3-tuple $(\mathcal{N}, \rightarrow, m_0)$, where

- $\mathcal{N}$ is a set of *nodes* ranging over *Method* names;
- $\rightarrow \subseteq \mathcal{N} \times \mathcal{N}$ is a set of *transitions*;
- m_0 is the initial state.

Notice that the transitions of this graph are not labelled and that there are no final states. Indeed, $\mathcal{M}(P)$ models the *method call* relationship: $m \rightarrow m'$ if and only if $\exists pc \cdot \mathit{InstrAt}(m, pc) = \texttt{invokevirtual}\ m'$. Whenever understood from the context we will write $\mathcal{M}$ instead of $\mathcal{M}(P)$. Given the method-call graph, we say that a method m' is *reachable* from m if and only if there is a path in the graph from m to m'. More formally, $\mathit{Reach}(m) = \{m' \mid m \rightarrow^+ m'\}$. Given the graph of method calls, we define the following predicate which determines whether a given method m is recursive:

$$\mathit{MutRec}(m) \equiv m \in \mathit{Reach}(m).$$

The following predicate characterises not only the mutually reachable methods but also the methods reachable from those:

$$\mathit{MutRecReach}(m) \equiv \exists m' \cdot \mathit{MutRec}(m') \wedge m \in \mathit{Reach}(m').$$

We introduce a *satisfaction* relation: $\mathcal{M} \models \phi$ iff $\mathcal{M}$ satisfies the property ϕ. We state now soundness and completeness of *Rec*.

Theorem 4 (Soundness and Completeness of *Rec*) *$\mathit{Rec}_{m,pc}$ if and only if $\mathcal{M} \models \mathit{MutRecReach}(m)$.* □

4.2.4 Main algorithm

We state now the soundness and completeness of our algorithm with respect to our abstraction, which follow directly from the definition of Γ and the soundness and completeness of *Loop*, *Loop*' and *Rec*.

Theorem 5 (Soundness of the algorithm) *If $\Gamma(m, pc) = \infty$, for some (m, pc), then $\mathit{InstrAt}(m, pc) =$ $\texttt{new}(cl)$ and such instruction occurs in a syntactic cycle and/or in a recursive method. Furthermore, if $\Gamma(m, pc) = 1$, for some (m, pc), then $\mathit{InstrAt}(m, pc) = \texttt{new}(cl)$ and $\mathit{InstrAt}(m, pc)$ is not in a syntactic cycle nor in a recursive method.*

Theorem 6 (Completeness of the algorithm) *If $\mathit{InstrAt}(m, pc)$ occurs in a syntactic cycle and/or in a recursive method and $\mathit{InstrAt}(m, pc) = \texttt{new}(cl)$, then $\Gamma(m, pc) = \infty$. On the other hand, if $\mathit{InstrAt}(m, pc) =$ $\texttt{new}(cl)$ does not occur in a syntactic cycle nor in a recursive method then $\Gamma(m, pc) = 1$.*

Notice that the above soundness and completeness result are with respect to an abstraction, which is the identification of `new` instructions inside *syntactic* cycles. However, the real interesting conclusion is:

Corollary 2 *If $\Gamma(m, pc) = 1$ then for any real execution of the applet, $\mathit{InstrAt}(m, pc)$ will be executed a finite number of times.* □

Our algorithm may be easily refined to give an upper bound for the memory used by an applet if no `new` instruction occurs inside a loop. This may be done by simply counting their occurrences and considering the memory allocated by each `new`, as done in [6].

$$\frac{(m, pc) : \texttt{throw}\ e \qquad (m, pc') \in \mathit{findHandler}(m, pc, e)}{F(\mathit{Loop}(m, pc)) \sqsubseteq \mathit{Loop}(m, pc')} \tag{16}$$

$$\frac{(m, pc) : \texttt{throw}\ e \qquad (m', pc') \in \mathit{findHandler}(m, pc, e) \qquad m' \neq m}{G(\mathit{Rec}(m, pc), m') \sqsubseteq \mathit{Rec}(m', pc')} \tag{17}$$

Table 4: Added rules for `throw`

$$\frac{(m, pc) : \texttt{jsr}\ pc'}{\begin{array}{l} F(\mathit{Loop}(m, pc)) \sqsubseteq \mathit{Loop}(m, pc') \\ F(\mathit{Loop}(m, pc)) \sqsubseteq \mathit{Loop}(m, pc + 1) \end{array}} \tag{18}$$

$$\frac{(m, pc) : \texttt{ret}\ i}{\bot \sqsubseteq \mathit{Loop}(m, \mathrm{END}_{ret})} \tag{19}$$

Table 5: Added rules for `jsr`

5 Handling exceptions and subroutines

One advantage of the rule-based approach presented in the previous section is its ease of extension. We sketch how to extend the rules for *Loop* and *Rec* in order to cover particular cases of exception handling and subroutine calls.

5.1 Exceptions

According to the operational semantics of the JCVM, an exception is raised either when a program instruction violates its semantic constraints, or when the current instruction is a `throw`. To illustrate the flexibility and modularity of the approach, we extend the algorithm of the previous section to handle exceptions raised by the `throw` instruction. Exceptions are slightly difficult to treat statically because finding the right exception handler may require searching through the frame stack. Obviously, this is not possible at compile time, hence we use a function $\mathit{findHandler}(m, pc, e)$ [16] that over-approximates the set of possible handlers that could possibly catch exception e when raised from location (m, pc).

The rules in Table 4 extend the constraints on *Loop* and *Rec* to handle explicit exceptions. Rule (16) takes care of the case when there is a handler for the exception in the current method. When there is a handler outside the current method, a `throw` is treated like a method invocation –rule (17).

5.2 Subroutines

The `finally` block of a `try...finally` Java construct is usually compiled into a subroutine, a fragment of code called with the `jsr` bytecode instruction. We treat subroutines as in Leroy's bytecode verifier [11], i.e. "as a regular branch that also pushes a value of type 'return address' on the stack; and `ret` is treated as a branch that can go to any instruction that follows a `jsr` in the current method". We assume that the applet being analysed has passed bytecode verification, guaranteeing the above property. Notice that the treatment of `ret` represents a considerable loss of precision since the analysis should take into account all the *pc*-numbers after the `jsr` instruction as possible return addresses of the subroutine. To simplify the presentation, in Table 5 we consider that the return address is always the instruction immediately after the `jsr` instruction (like in method calls); END_{ret} may be defined similarly as END_m. The more general case described by Leroy could easily be represented with the help of an auxiliary set function, analogous to *findHandler* (cf. Section 5.1).

6 Final discussion

This work is a first step in the construction of a certified, on-card analyser for estimating memory usage on Java cards. We have given a constraint-based algorithm which detects all the potential loops and recursive methods in order to determine the presence of dynamic class instantiations inside cycles. We have proved its soundness and completeness w.r.t. an abstraction of the operational semantics. Our abstraction is conservative and correct w.r.t. a run-time execution of the program, in the sense that if a run-time cycle exists,

then such cycle is detected by our algorithm, and if our analysis gives as an answer that no `new` instruction is inside a cycle, then it is indeed the case. Besides the advantages mentioned in the introduction, the modularity of our algorithm allows the analyser, as well as the functions *Loop*, *Loop*' and *Rec*, to be reused by other constraint-based analysers as programs which have been proved correct.

We have defined two functions, *Loop* and *Loop*' for detecting cycles. Notice that we could have defined only *Loop* just changing the rule of `invokevirtual` to:

$$\frac{(m, pc) : \texttt{invokevirtual}\ m'}{\begin{array}{l} \mathit{Loop}(m, pc) \sqsubseteq \mathit{Loop}(m', 1) \\ \mathit{Loop}(m, pc) \sqsubseteq \mathit{Loop}(m, pc + 1) \end{array}}$$

In this case we would not need the definition of *Loop*'. Our choice is, however, not arbitrary and it is motivated by modularity and memory usage concerns. Computing the intra-procedural cycles first yields a local reasoning which may be done once and for all, also minimising the auxiliary memory used.

Complexity of the analysis. Let N be the number of bytecode instructions, N_U the number of unconditional instructions (including `throw`) and N_C the number of conditional jump instructions in a given method m. If B is the number of bits used for representing a *pc*-number, then the memory needed to compute *Loop* using the algorithm in Section 3.1 is bounded by $N \times ((N_U + 2N_C + 1) \times B + 1)$. The second factor is a bound on the size of $\mathit{Loop}(m, pc)$ for a fixed *pc*, where $(N_U + 2N_C + 1)$ estimates the number of basic blocks in method m: besides the first basic block, each unconditional jump instruction may determine one basic block, and each conditional jump may determine two basic blocks. An instruction found to be in a potential cycle needs one further bit (corresponding to $\bullet$). This information is propagated along the method, and in the worst case to all of its instructions. For *Rec* the auxiliary memory used is definitely less, since we only propagate method's names, which might be associated to each method without needing to propagate them to every method line, although the specification presented here does it. The time efficiency of our algorithm strongly depends on the efficiency of the constraint solver, but we believe that in principle each fix-point computation converges in at most three iterations. Moreover, the order in which the different predicates are applied may drastically improve the performance and space-complexity of the analyser. By computing *Rec* first, for instance, we would not need to compute *Loop* for the methods already marked as recursive.

A moderately complex Java Card method may have around 50 basic blocks (see [11, Section 2.3]). Considering a method with up to 200 instructions and 50 basic blocks the computation of *Loop* would require approximately 10KB of memory in the worst case. This still exceeds the 4 KB of RAM available in top-quality smart cards, so it is not yet feasible to deploy the analyser on-card. There is possibly enough transient memory (EEPROM) to store the data structures used by the analyser, but its latency (1-10 ms per write operation) would make the analysis extremely slow. On the other hand, the analysis could certainly fit in, for instance, Java-enabled mobile phones.

Related Work. Ad-hoc type systems are probably the most successful tools for guaranteeing that well-typed programs run within stated space-bounds. Previous work along these lines can be found in the context of typed assembly [2, 17] and functional languages [3, 8, 9]. In [2], the authors present a first-order linearly typed assembly language which allows the safe reuse of heap space for elements of different types. The idea is to design a family of assembly languages which have high-level typing features (e.g. the use of a special *diamond* resource type) which are used to express resource bound constraints. Vanderwaart et al. [17] describe a type theory for certified code, with the aim of guaranteeing bounds on CPU usage, by using an instruction-counting mechanism, or "virtual clock". The approach may be extended to ensure bounds on other resources as well. Another recent work is [1] where the resource bounds problem is studied on a simple stack machine. The authors show how to perform type, size and termination verifications at the level of the bytecode. In [13] it is shown how to import ideas from data flow analysis, using abstract interpretation, into inter-procedural control flow analysis. Even though the motivations and the techniques are not the same, it is worth mentioning the work done by the Mobile Resource Guarantees project [12], which applies ideas from proof-carrying code for solving the problem of resource certification for mobile code.

We took inspiration from [5] where a similar technique is applied with success to extract a data flow analyser for Java card bytecode programs. More recently, a Coq-certified constrained-based memory usage analyser has been presented in [6], also based on the formalism introduced in [5]. The analyser has been automatically obtained from its correctness proof using Coq's extraction mechanism. Although such algorithm has simpler rules for computing *Loop*, the space complexity is higher than the one presented here.

To compute *Loop* we have chosen to keep the *pc*-numbers of the target of any jump whereas [6] keeps the *pc*-numbers of all the instructions already visited. Given an applet of 200 lines with 50 basic blocks (including exception handlers) the auxiliary memory for computing *Loop* (in the worst case) using the algorithm defined in [6] would reach about 40 KB, contrasting with the 10 KB of our algorithm.

Future Work. Due to the inter-procedural dependencies the analysis is not compositional: if two applets are "loop-safe" (i.e., no `new` occurs inside a cycle), their composition is not necessarily loop-safe. We should, in principle, compute *Rec* again for the composite applet in order to guarantee the non-existence of `new` instructions inside newly created recursive methods. However, it is possible to minimise the computation of a global fix-point (for detecting recursive methods) by keeping relevant information regarding the methods of one applet called by methods of the other (and vice-versa). Our analyser may be easily extended in that direction.

In Section 5 we have given an idea of how to extend our rules in order to handle exceptions and subroutines. We intend to extend our approach to cover the remaining cases of exception handling and to improve the precision of the subroutine analysis.

The space complexity of the intra-procedural analysis may be improved by propagating the *pc*-numbers only to the beginning of a basic block instead of to all the internal instructions. In this case the memory required would be bounded by $((N_U + 2N_C + 1)^2 \times B) + (N_U + 2N_C + 1)$; on an applet with 50 basic blocks (and at most 256 instructions), it would need only 2.5 KB, leaving on-card verification at reach.

Acknowledgements. The second author is indebted to Thomas Jensen and David Pichardie for insightful discussions in an early stage of this work. Patrick Bernard, Eduardo Giménez, Arnaud Gotlieb and Jean–Louis Lanet have provided us with useful information on the Java card technology.

References

[1] R.M. Amadio, S. Coupet-Grimal, S. Dal Zilio, and L. Jakubiec. A Functional Scenario for Bytecode Verification of Resource Bounds. Research report 17-2004, LIF, Marseille, France, 2004.

[2] D. Aspinall and A. Compagnoni. Heap-bounded assembly language. *J. Autom. Reason.*, 31(3-4):261–302, 2003.

[3] D. Aspinall and M. Hofmann. Another type system for in-place update. In *ESOP*, volume 2305 of *LNCS*, pages 36–52, 2002.

[4] Y. Bertot and P. Casteran. *Interactive Theorem Proving and Program Development. Coq'Art : the Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer Verlag, 2004.

[5] D. Cachera, T. Jensen, D. Pichardie, and V. Rusu. Extracting a data flow analyser in constructive logic. In *ESOP*, LNCS, pages 385–400, 2004.

[6] D. Cachera, T. Jensen, D. Pichardie, and G. Schneider. Certified memory usage analysis. In *Formal Methods*, volume 3582 of *LNCS*, pages 91–106, 2005. To appear.

[7] Zhiqun Chen. *Java Card technology for Smart Cards: architecture and programmer's guide*. Java series. Addison Wesley, 2000.

[8] M. Hofmann. A type system for bounded space and functional in-place update. *Nordic Journal of Computing*, 7(4):258–289, 2000.

[9] J. Hughes and L. Pareto. Recursion and dynamic data-structures in bounded space: Towards embedded ML programming. In *International Conference on Functional Programming*, pages 70–81, 1999.

[10] X. Leroy. Java bytecode verification: an overview. In *CAV'01*, volume 2102 of *LNCS*, pages 265–285. Springer-Verlag, 2001.

[11] X. Leroy. Bytecode verification on java smart cards. *Softw. Pract. Exper.*, 32(4):319–340, 2002.

[12] MRG. Mobile Resource Guarantees project. See `http://groups.inf.ed .ac.uk/mrg/`.

[13] F. Nielson and H.R. Nielson. Interprocedural control flow analysis. In *European Symposium on Programming*, pages 20–39, 1999.

[14] F. Nielson, H.R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer-Verlag New York, Inc., 1999.

[15] D.A. Schmidt and B. Steffen. Program analysis as model checking of abstract interpretations. In *SAS*, LNCS, pages 351–380, 1998.

[16] I. Siveroni. Operational semantics of the Java Card Virtual Machine. *J. Logic and Algebraic Programming*, 58(1-2), 2004.

[17] J.C. Vanderwaart and K. Crary. Foundational typed assembly language for grid computing. Technical Report CMU-CS-04-104, CMU, February 2004.

Un Editor de Programas por Refinamiento

Ascánder J. Suárez, Wendi C. Urribarrí, Vicente Yriarte
Universidad Simón Bolívar,
Departamento de Computación y Tecnología de la Información,
Caracas, Venezuela, 89000
suarez@ldc.usb.ve, wendi@ldc.usb.ve, yriarte@ldc.usb.ve

Abstract

A tool is presented with the purpose of helping the user to develop programs from specifications. This transformation is carried out by means of a sequence of refinement steps that guarantee that the final program satisfies the original specification. The language on which the programs are written is a subset of *GaCeLa*—inspired in Dijkstra's *guarded command language* and variants of Hoare triples—, extended so that it admits Morgan's specification statements. To carry out the transformations, the *GaCeLa* programs are represented as XML trees, and the transformations are implemented as templates in XSLT that transform a XML document representing the program to be refined into another XML document that represents the refined program. The tool we present is interactive at a high level, flexible or not restrictive (it does not impose a derivation style on the user) and maintains a history of the applied refinement steps. In the transformations we use a subset of the transformation rules proposed by Morgan. We do not discuss details of the correctness of these rules here.

Keywords: Program refinement, Formal specification, Program transformation, Formal methods, Program derivation.

Resumen

Se presenta una herramienta cuyo fin es ayudar al usuario a desarrollar programas a partir de especificaciones, mediante una secuencia de pasos de refinamiento que garanticen que el programa final satisface la especificación original. El lenguaje sobre el que se escriben los programas es un subconjunto del lenguaje *GaCeLa*— que está inspirado en el *lenguaje de comandos con guardias* propuesto por Dijkstra y en variantes de las tripletas de Hoare—, extendido para que admita especificaciones al estilo Morgan. Para llevar a cabo las transformaciones, los programas originalmente escritos en *GaCeLa* se representan como árboles XML, y las transformaciones se implementan como plantillas en XSLT, que transforman un documento XML que representa el programa que se quiere refinar en otro documento XML que representa el programa refinado. La herramienta presentada es interactiva a alto nivel, flexible o no restrictiva (no impone al usuario un estilo de derivación) y mantiene una historia de los pasos de refinamiento aplicados. En las transformaciones usamos un subconjunto de las reglas de transformación discutidas por Morgan. No discutimos aquí los detalles de corrección de dichas reglas.

Palabras claves: Refinamiento de programas, Especificación formal, Transformación de programas, Métodos formales, Derivación de programas.

1 Introducción

La idea de derivar programas usando transformaciones ha sido abordada por muchos investigadores [2, 7, 11, 9], pero los resultados prácticos obtenidos han sido escasos [10], debido a que la derivación a mano de casi cualquier programa está llena de largas frases de manipulación formal que son tediosas y susceptibles de equivocaciones. Es por ello que la utilidad de los métodos de derivación de programas dependen de que exista un soporte automatizado a dicho proceso de manipulación.

En este trabajo presentamos un editor de especificaciones y programas cuya finalidad es automatizar todo lo que se pueda el proceso de deducción de un programa ejecutable a partir de su especificación. El mismo tiene carácter de prototipo, en el sentido de que sólo implementa parte de las funcionalidades que pudiera poseer y en que será usado para producir una herramienta más completa.

Nuestro editor fue diseñado para que satisficiese al menos los siguientes principios, algunos de ellos mencionados en [10]:

- Ser interactivo. Dejar que el usuario tome las *decisiones de diseño* y manejar los detalles formales.
- La interacción con el editor debe ser de suficiente *alto-nivel*.
- El editor debe ser flexible. No debe imponer al usuario una disciplina demasiado restrictiva sobre la forma en que los programas se deben derivar.
- Proveer manipulación de archivos: abrir, guardar, cerrar.
- Debe poder mostrar la lista de las reglas de refinamiento aplicables a la instrucción que se quiere refinar y permitir la selección de alguna.
- El sistema debe ser capaz de mantener un historial de las transformaciones realizadas.

Como lenguaje de programación usamos una extensión de un subconjunto de *GaCeLa* [13], que es un lenguaje inspirado en el *Lenguaje de Comandos con Guardias* de Dijkstra [3, 4] y en las tripletas de Hoare [5]. *GaCeLa*, desarrollado por Ascánder Suárez, permite verificar condiciones de corrección de un programa, y fue extendido para que permita incluir especificaciones al estilo de Morgan [8, 9]. Así que los programas en este lenguaje están formados por instrucciones ejecutables y especificaciones, tal como se aclarará más adelante.

En la fase de transformación de los programas usamos reglas de refinamiento estudiadas por Morgan que se implementaron como plantillas de transformación. Para aplicar una *regla de transformación* a un programa en *GaCeLa*, éste se transforma en un árbol XML y sobre éste se aplica la regla que se implementa como una plantilla XSLT. Esta plantilla transforma un documento XML que representa el programa que se quiere refinar, en otro documento XML que representa el programa refinado. Finalmente este documento XML se transforma de nuevo en un programa en *GaCeLa* para ser mostrado.

El lenguaje de marcado extensible (XML *Extensible Markup Language*) [14] es un conjunto de reglas que permite estructurar los datos de un documento usando marcas. Para usarlo es necesario, en primer lugar identificar todos los tipos de información que se incluirán en el documento, y en segundo lugar, proporcionar una definición de la estructura que rodea a cada pieza de información que se identificó en el documento. Esto se hace con una *definición de tipo de documento* (Document Type Definition DTD), cuyo fin es describir la estructura de los contenidos usadas en el documento XML. Debido a que se pueden usar identificadores y descripciones de contenido propias, se puede usar XML para describir cualquier tipo de información que se pueda clasificar en tipos específicos de contenido.

El lenguaje de estilo extensible (XSL Extensible StyleSheet Language) [14] es un término usado para englobar dos estándares propuestos por el *World Wide Web Consortium* (W3C). Ellos permiten que se pueda expandir el uso de los documentos XML. XSL proporciona una estructura para convertir documentos XML a otros formatos de texto estructurado, y permite dar estilo a documentos XML para su presentación en diversos medios. El primero de ellos se denomina XSLT por *Transformaciones XSL*.

El editor permite seleccionar una parte del programa para aplicarle una transformación. El sistema determina automáticamente qué transformaciones se pueden aplicar en base a las condiciones de aplicabilidad

de las reglas. Si la regla tiene alguna condición lógica que se deba satisfacer para que se pueda aplicar, se añade al programa resultante una *obligación de prueba* (para ello se usan las coerciones de Morgan). Estas obligaciones se pueden verificar a mano o con la ayuda de un demostrador automatizado como ACL2 [6].

El insumo original del editor es o bien un programa abstracto—una especificación formal al estilo de Morgan donde básicamente se indica la *precondición*, la *post-condición* y un *marco* con las variables que se permite modificar—o un programa parcialmente desarrollado de una sesión anterior.

El editor permite mantener la historia de las transformaciones ejecutadas hasta los momentos y permite deshacer pasos previos, por ejemplo, los que no condujeron a un resultado satisfactorio.

Esta herramienta no sólo permite deducir programas correctos sino que también le permite al usuario la posibilidad de pasearse sobre posibles variantes de la solución del problema con simplemente hacer una selección diferente de la regla a aplicar en un determinado momento. El hecho de que el programador no tenga que estar chequeando a mano detalles mecánicos le da la oportunidad de estudiar en mayor profundidad su problema y obtener un algoritmo más eficiente.

El proceso de desarrollo de este editor se llevó a cabo por medio de una secuencia de etapas en las que se fueron implementando las partes. A continuación se presenta una lista cronológica de las tareas realizadas.

- Definición del lenguaje de especificación y programación que se usa para escribir los programas y especificaciones.
- Desarrollo de un reconocedor para el lenguaje de especificación y programación definido.
- Selección de las reglas de refinamiento que se desarrollaron.
- Definición y desarrollo de plantillas XSL para las reglas de refinamiento.
- Desarrollo de plantillas XSL para las condiciones de aplicabilidad de las reglas de refinamiento.
- Definición de la representación de la *historia del refinamiento*.
- Creación de las ventanas de la interfaz con el usuario.
- Integración del reconocedor del lenguaje, de las plantillas validadoras de la aplicabilidad de las reglas y de las reglas de refinamiento, y del mecanismo de carga y de descarga de la historia, con las ventanas de interfaz con el usuario.

El resto de este artículo se organiza como sigue: en la siguiente sección presentamos el fundamento teórico del paradigma de derivación de programas vía transformaciones o refinamiento. En la sección 3 presentamos brevemente el lenguaje de programación que maneja el editor. En la 4, se describe el editor por etapas. Primero se abordan detalles técnicos como reconocimiento del lenguaje y la forma como se llevan a cabo las transformaciones, para terminar mostrando cómo usar el editor. Finalmente, en la sección 5 se presentan algunas conclusiones y comentarios finales sobre el trabajo.

2 Especificaciones y Refinamiento

Una especificación es una descripción formal de lo que un programa debe hacer, que indica de manera precisa las condiciones que se deben cumplir al inicio y al final del programa. La precisión se debe a que usa el lenguaje matemático para expresar el comportamiento y las condiciones que deben cumplir el estado inicial y el final. También puede verse como un programa abstracto, que si bien no lo puede ejecutar una computadora, es más comprensible que un programa ejecutable porque en el proceso de especificación de un programa los requerimientos se escriben de manera más clara. Los programas ejecutables que usualmente son llamados código suelen no ser claros, en especial los grandes [9]. Otros beneficios que trae especificar es que ayuda a descubrir fallas de diseño, descripciones incompletas e inconsistencias en la definición de los componentes del programa.

En el proceso de desarrollar un programa a partir de una especificación se pueden generar varias subespecificaciones, de manera que la construcción del programa se puede dividir por componentes. Esto permite

que se pueda estructurar mejor el diseño del programa que se desea desarrollar, y también, que se pueda simplificar el proceso de verificación de la corrección. Además, mejora la comunicación entre los actores que actúan en la producción de programas grandes, como son el cliente, el diseñador, el programador y el *que lo debe probar*. Refraseando a Broy, "una especificación formal es un requisito previo para una prueba formal y una base valiosa para el desarrollo de programas" [1]. Estas observaciones justifican la propuesta de incluir a las especificaciones como una acción más dentro de los programas y no tratarlas como construcciones distintas.

Morgan introduce las especificaciones como una extensión al Lenguaje de Comandos con Guardias de Dijkstra, con algunas variantes, y utiliza esa nueva versión del lenguaje en su propuesta sobre especificaciones y refinamiento [8, 9]. La construcción de especificación que plantea es una instrucción no ejecutable que consiste de una lista m de variables y dos predicados pre y $post$ sobre el conjunto de las variables del programa:

$$m\colon [pre, post] \tag{1}$$

donde:

- el marco m de variables es una lista, que puede ser vacía, de las variables que se permite cambiar
- el predicado pre es una fórmula que describe el conjunto de posibles estados iniciales de la especificación
- el predicado $post$ es una fórmula que describe el conjunto de posibles estados finales de la especificación

Informalmente, la instrucción de especificación significa: asumiendo un estado inicial que satisface pre, entonces se establece un estado final que satisfaga $post$ cambiando solamente variables que pertenezcan al marco m [8]. Si la precondición es el predicado $true$, que es cierto cualquiera sea el estado, se puede omitir la precondición y en lugar de escribir $m : [true, post]$ se escribe $m : [post]$. Si el marco es vacío, la especificación se escribe $[pre, post]$. Las especificaciones de la forma $[pre, true]$ se abrevian como $\{pre\}$, y se llaman aserciones. Las aserciones o no terminan porque su precondición no se cumple, o terminan pero no cambian el estado del programa pues su marco es vacío [9]. Se introducen en varios puntos de un programa para verificar que cierta condición se cumple antes de ejecutar la acción que les sigue en el programa. Las especificaciones de la forma $[true, post]$ se abrevian $[post]$, se llaman coerciones y se comportan como *skip* cuando $post$ es $true$. De lo contrario, se comportan como el programa infactible *magic*, el cual es una especificación que no se puede refinar a ningún código, que siempre termina y establece un estado final deseado aunque esto sea imposible.

Se han propuesto varias metodologías para el desarrollo de programas a partir de una especificación formal. Nos interesa la metodología de transformación de programas por refinamiento.

El *refinamiento* es un proceso en el que se toma un programa y, mediante la aplicación de reglas que preserven la corrección, se transforma en otro que se espera esté más cerca de ser ejecutable que el anterior. Las especificaciones son programas abstractos; a los programas que se pueden ejecutar se les suele denominar *código*. También se llama código a las instrucciones que la computadora puede entender y que usa como base para realizar acciones. Se puede decir que después de un paso de refinamiento el programa obtenido tiene menos nivel de abstracción que el anterior. La transformación de programas de lo abstracto a lo concreto está mediada por la relación $\sqsubseteq$ de refinamiento que se define como sigue: Se dice que el programa Q refina al programa P y se escribe $P \sqsubseteq Q$ si y sólo si toda especificación que P satisface, también Q la satisface. Esto es,"Q es un reemplazo aceptable para P" [8].

Morgan propuso un sistema de leyes para el refinamiento de especificaciones y transformación de programas [9], en el que se basa para presentar una metodología de desarrollo de programas a partir de especificaciones. En el editor se utiliza un subconjunto de estas reglas para refinar las especificaciones y los programas intermedios. A continuación mostramos dos de esas reglas de refinamiento:

$$\begin{array}{rcl} [pre \;\Rightarrow\; post[w\backslash E]] & \Longrightarrow & w, x\colon [pre, post] \;\sqsubseteq\; w := E \\ [pre \Rightarrow \exists i.G_i] & \Longrightarrow & w\colon [pre, post] \;\sqsubseteq\; \textbf{if } [\!]i.G_i \;\rightarrow\; w\colon [G_i \;\wedge\; pre, post] \textbf{ fi } . \end{array} \tag{2}$$

Para ilustrar la técnica de derivación mediante refinamiento consideremos el problema de calcular el máximo entre dos enteros m y n. Escrita en nuestro lenguaje de especificación y programación, que describiremos brevemente en la siguiente sección, la especificación inicial luce como:

$$
\begin{array}{l}
program\ maximo\ |[\\
\quad const\ m = readInt(\text{"}n?\ \text{"}), n = readInt(\text{"}m?\ \text{"}) : int \\
\quad ; var\ r := 0 : int \\
\quad ; r{:}\,[r = n\ max\ m]\ (E) \\
\quad \{r = n\ max\ m\} \\
]|
\end{array}
$$

Puesto que $true \Rightarrow m \leq n \vee m \geq n$, si tomamos como guardias $m \leq n$ y $m \geq n$, podemos usar la **regla de alternación**—segunda de las reglas de (2)—para obtener el siguiente programa refinado.

$$
\begin{array}{l}
program\ maximo\ |[\\
\quad const\ m = readInt(\text{"}n?\ \text{"}), n = readInt(\text{"}m?\ \text{"}) : int\ ; \\
\quad var\ r := 0 : int\ ; \\
\quad [\ true ==> n >= m \vee n <= m\ ]\ (C1)\ ; \\
\quad if\ n >= m ->\ r{:}\,[n >= m, r = n\ max\ m]\ (E2) \\
\quad [\!]\, n <= m ->\ r{:}\,[n <= m, r = n\ max\ m]\ (E3) \\
\quad fi \\
\quad \{r = n\ max\ m\} \\
]|
\end{array}
$$

La especificación E se transformó en un condicional que contiene dos especificaciones $E2$ y $E3$ que debemos refinar y se agregó la especificación $C1$ que es una coerción que representa la obligación de corrección generada al aplicar la regla de la alternación, en la que la precondición de la especificación debe implicar a la disyunción de los guardias. La especificación $E2$ tiene como precondición que $n \geq m$ y como postcondición que $r = n \max m$. Como r se puede modificar, y es fácil ver que la precondición implica a la postcondición con r sustituido por n, se puede aplicar la regla de **asignación** (primera parte de 2). Razonando de manera similar para $E3$ se tiene:

$$
\begin{array}{l}
program\ maximo\ |[\\
\quad const\ m = readInt(\text{"}n?\text{"}), n = readInt(\text{"}m?\text{"}) : int\ ; \\
\quad var\ r := 0 : int\ ; \\
\quad [\ true ==> n >= m \vee n <= m\ ]\ \ (C1)\ ; \\
\quad if\ n >= m ->\ \ [\ n >= m ==> n = n\ max\ m\ ]\ \ (C2)\ ; r := n \\
\quad [\!]\ n <= m ->\ \ [\ n <= m ==> m = n\ max\ m\ ]\ (C3)\ ; r := m \\
\quad fi \\
\quad \{r = n\ max\ m\} \\
]|
\end{array}
$$

Es fácil ver que las condiciones de las coerciones $C1$, $C2$ y $C3$ son tautologías, lo cual prueba que el programa derivado es correcto.

3 Lenguaje de Especificación y Programación

El lenguaje L de especificación y programación que se diseñó para usar en el editor es una extensión de un subconjunto del lenguaje *GaCeLa*, el cual se usa actualmente para la enseñanza de las asignaturas de algoritmos, en la Universidad Simón Bolívar, debido a su capacidad de permitir agregar condiciones de corrección que se verifican dinámicamente y de producir las obligaciones de corrección del programa. *GaCeLa* se basa en el Lenguaje de Comandos con Guardias de Dijkstra y en las tripletas de Hoare. El lenguaje L está formado por las acciones y subprogramas de *GaCeLa* y especificaciones. En L, (si no consideramos las aserciones y las coerciones), las especificaciones se pueden ver como *promesas* de construcción de un programa que satisfaga lo que establecen. No son programas ejecutables, sino programas abstractos. La intención es transformarlos en programas concretos ejecutables, esto es, en código. Un programa escrito en el lenguaje L, al igual que en *GaCeLa*, tiene un nombre o identificador, puede tener definido dentro de él subprogramas (funciones y procedimientos), y siempre tiene un cuerpo formado por una secuenciación de tripletas de Hoare extendidas

$$\{P\}^*\ R\ \{Q\}^* \tag{3}$$

donde $\{P\}^*$ y $\{Q\}^*$ son secuencias de cero o más aserciones, y R es una acción; las tripletas van separadas por el símbolo " ; ". Por ejemplo, un programa (*abstracto*) que permita sumar dos números enteros positivos escrito en L podría lucir como sigue (vea también los ejemplo de la sección anterior):

$$
\begin{array}{l}
program\ suma\ |[\\
\quad const\ A = readInt("A?") : int\ \{A >= 0\} \\
\quad ;\ const\ B = readInt("B?") : int\ \{B >= 0\} \\
\quad ;\ var\ x := 0 : int \\
\quad ;\ x{:}\,[true, x = A + B] \\
\quad \{x = A + B\} \\
]|
\end{array}
$$

3.1 Especificaciones

La sintaxis de una especificación es similar a la sintaxis de las especificaciones de Morgan: $m{:}\,[pre, post]$ donde m es una lista de las variables que se permite modificar y se denomina *marco*, pre es la precondición y $post$ la postcondición. Tanto el marco como la precondición pre, pueden no escribirse. En el primer caso, no se permite cambiar ninguna variable y en el segundo, se sobreentiende que pre es $true$. Es obligatorio poner la postcondición. Esta especificación es un programa abstracto que estable que si se parte de un estado que satisface la precondición pre, sólo modificando a las variables en m se termina en un estado que satisface la postcondición $post$. Al caso particular $[pre, true]$, donde $true$ representa el predicado que se satisface para todos los estados, se escribe por $\{pre\}$ y se corresponde con las aserciones de *GaCeLa*. Su interpretación es: *pre se debe satisfacer en este punto del programa, sin cambiar nada* .

3.2 Instrucciones

Las instrucciones de L son la *especificación*, ya descrita en la sección anterior más un subconjunto de las acciones de *GaCeLa*, formado por: **skip**, declaración, asignación múltiple, llamadas a procedimientos, alternación, iteración y bloques locales. La semántica de estas instrucciones, exceptuando la de los bloques locales, es igual que en el lenguaje de Comandos con Guardias de Dijkstra, pero *GaCeLa*, a diferencia del lenguaje de Comandos con Guardias, tiene bloques locales que no necesariamente deben tener una declaración cuando se abren. Además puede declararse una variable o constante en cualquier punto dentro de un bloque. Las variables se inicializan en la declaración.

4 El Editor

En esta sección presentaremos la descripción del editor en tres etapas. En la primera se describe cómo se representan los programas que se van a transformar. En la segunda se describe el proceso de aplicación de las transformaciones. Y finalmente en la tercera se describe cómo se usa el editor y su interfaz con el usuario.

4.1 Representación del Lenguaje

Para reconocer el lenguaje L descrito en la sección 3 se construyó un reconocedor usando el generador de analizadores *Claire 4.71* [12], desarrollado en la Universidad Simón Bolívar, usando el algoritmo LALR(1) y definiendo las acciones semánticas en el lenguaje Java. El reconocedor obtenido produce toda la información sobre la estructura de la frese leída, necesaria para el desarrollo del editor.

Después que un programa ha sido reconocido por el analizador se transforma a una representación arborescente cuyos hijos son las subexpresiones que lo componen. Cada componente del lenguaje tiene asociada operaciones que permiten obtener su representación como string y como árbol XML.

Para describir los programas del editor, se definió un DTD en el que se describe con exactitud cómo deben estar estructuradas cada una de las partes del programa y los componentes de dichas partes. En la representación se incluye un nombre que sirve para identificar el nodo del árbol XML que se desea transformar. Por ejemplo, la representación XML de la especificación $x{:}\,[true, x = A + B]$ del programa *suma* que

se mostró en la sección 3 es

```
< ESPECIFICACION nombre = 'SUM' charStart = '101' charEnd = '128' >
    < MARCO >
        < VARIABLE nombre = 'x'/ >
    < /MARCO >
    < PRECONDICION >
        < CONSTANTE tipo = 'BOOLEANO' valor = 'true'/ >
    < /PRECONDICION >
    < POSTCONDICION >
        < OPERADOR nombre = ' ='>
            < VARIABLE nombre = 'x'/ >
            < OPERADOR nombre = '+' >
                < VARIABLE nombre = 'A'/ >
                < VARIABLE nombre = 'B'/ >
            < /OPERADOR >
        < /OPERADOR >
    < /POSTCONDICION >
< /ESPECIFICACION >
```

cuya regla de construcción está definida en el DTD del lenguaje L

```
<ELEMENT ESPECIFICACION ((MARCO?, PRECONDICION?, POSTCONDICION)
                         | PRECONDICION) >
<ATTLIST ESPECIFICACION nombre NMTOKEN #IMPLIED
                        charStart NMTOKEN #REQUIRED
                        charEnd NMTOKEN #REQUIRED >
```

Nótese que los programas tienen tres representaciones: como texto, utilizado para desplegarlo, como árbol en Java y como árbol en XML.

4.2 Transformaciones

Para manejar el proceso de transformación de los programas se usó XSLT que es un sub-lenguaje de XSL en el que se describe cómo transformar un documento en otro. Conviene ver estos documentos como árboles. Para realizar la transformación se debe especificar qué nodos del árbol fuente se desean modificar y cómo. XSL realiza la transformación en varios pasos:

- XSL estudia el árbol XML fuente y localiza los elementos que serán formateados y preparados para su presentación.
- XSL produce un árbol secundario que asigna el estilo especificado a los elementos XML.
- Un procesador XSL le aplica el estilo apropiado a los elementos y produce el documento en el nuevo formato.

4.2.1 Reglas de refinamiento

Básicamente, se tomó cada una de las reglas que se iban a implementar y se estudió cómo hacer una transformación del componente de programa que se desea refinar visto como un árbol XML. Cada regla se implementó como una plantilla escrita en el lenguaje de transformaciones XSLT, que permite darle nombre, asignarle un nivel de prioridad de aplicación e indicar cuál es el componente de programa —y las características que debe tener como mínimo—que se puede transformar usando esa regla de refinamiento.

Cada regla de transformación se implementa como un elemento $xsl : template$ de XSLT. Este elemento está formado por una parte sintáctica, donde se indica cuál es la estructura que debe tener el nodo del árbol XML fuente al que se le puede aplicar la regla, es decir, una expresión de correspondencia que indica qué porción o porciones del documento va a procesar la regla, y de una parte de definición de la estructura por la que se va a reemplazar el nodo o las acciones que se realizan cuando se encuentra una porción del documento fuente que corresponda con la expresión. La expresión de correspondencia se especifica por medio del atributo *match* del elemento $xsl : template$ [14]. Estos pueden necesitar parámetros que se pasan a la plantilla de la regla cuando se utiliza.

Las condiciones que se deben cumplir para que la aplicación de una regla de refinamiento sea correcta, se incluyen en el árbol resultante de la transformación como la representación en XML de lo que sería una *coerción*. Como no se cuenta con un motor de verificación que pueda automatizar las pruebas de esas condiciones (está en desarrollo), éstas quedan en el código del programa como coerciones, que luego se podrían eliminar cuando se prueben. Luego de realizada la transformación y obtenido el árbol resultante, éste se puede convertir a la forma de texto usando un método en Java, definido para ello en el editor.

4.2.2 Condiciones de aplicabilidad

Además de las plantillas de las reglas de refinamiento, también se implementaron unas plantillas que "verifican" si una regla es aplicable o no al árbol que representa al componente que se desea refinar. Estas plantillas tienen en su parte de reemplazo una marca que indica que la regla es aplicable y que se retorna como árbol resultado si se hace "match" con el nodo del árbol al que se le aplica la plantilla. Para cada una de las plantillas de las reglas de refinamiento existe otra plantilla de verificación de su aplicabilidad.

4.2.3 Representación de la historia del refinamiento

La historia del refinamiento es la secuencia de los cambios hechos sobre un programa cuando este se somete al proceso de refinamiento. Esta historia mantiene el estado inicial del programa—texto del programa—, la información relevante de cada transformación (nombre del programa en el que se aplicó, nombre del subprograma en el que se aplicó si fuera el caso, nombre de la especificación a la que se le aplicó si es el caso, nombre de la regla de refinamiento aplicada, fecha y hora de aplicación y la lista de parámetros que se utilizaron para aplicar la regla) y el estado final del programa. Si se viera como un grafo dirigido, los estados serían nodos y las transiciones serían los pasos de transformación, desde un nodo que representa el estado inicial a un nodo que representa el estado final.

Además, la historia debe guardar el estado del programa en caso de que se haya realizado un cambio en el texto que no haya sido por transformación mediante refinamiento, lo cual no es lo que se espera, pero que pudiera ocurrir, si el usuario inserta, elimina o modifica el programa.

Los estados y las transformaciones son representados como objetos en Java que se pueden traducir a árboles XML con la tecnología XML de Java mediante unos métodos definidos en sus clases. Al crear la estructura de la historia como un árbol XML, éste se puede transformar a la forma de *string* con las marcas o etiquetas bien formadas en XML y de acuerdo a un DTD definido.

4.3 Interfaz con el Usuario

La interfaz del editor es un conjunto de ventanas desarrolladas usando componentes Swing de Java. Swing provee un conjunto de componentes que se comportan de manera similar en todas las plataformas. La interfaz del editor está disponible en dos idiomas, inglés y castellano. El idioma por defecto depende de la configuración de la máquina. Para agregar otros idiomas sólo hace falta proveer una nueva tabla con los textos a mostrar. A continuación mostramos brevemente cómo se usa el editor.

Cuando se invoca al editor, aparece una ventana cuyo nombre es *"Editor de Especificaciones"* que muestra un menú principal y un panel de "pestañas". El menú principal del editor permite ejecutar varias opciones: de archivo, de proyecto y de ayuda. Las opciones de archivo nos permiten crear, abrir, guardar y cerrar archivos. Al crear un nuevo archivo aparece un cuadro de diálogo en el que se solicita los elementos de la especificación inicial. A continuación se muestra la pestaña de edición con la versión inicial del programa. Las opciones de proyecto, permiten ejecutar los comandos para manipular los programas; ellas son indentar, compilar, refinar y ver la historia. Cuando se pide refinar, se invoca al método para revisar si existen reglas de refinamiento aplicables a la especificación u otra construcción sobre la que está ubicado el cursor. Si las hay, aparece una ventana mostrando las leyes aplicables, que permite seleccionar una regla aplicable y ejecutar la aplicación. Si no, indica que no hay leyes aplicables en un cuadro de diálogo que permite abrir una ventana donde se muestran todas las reglas de refinamiento y su información asociada. Sobre la historia del refinamiento se pueden ejecutar tres comandos: deshacer, que permite deshacer los pasos

de refinamiento realizados; rehacer, que permite rehacer pasos de refinamiento realizados sobre el programa y ver historia, que abre una pestaña donde se muestra el estado del programa cuando se editó para transformarlo y las transformaciones hechas. En caso de que se haya realizado un cambio manual sobre el programa, también se muestra ese cambio. Se muestra la fecha y la hora de ejecución de cada transformación.

Nota: en esta versión prototipo sólo se permite trabajar con un archivo a la vez. Además los archivos deben contener programas escritos en el lenguaje L.

4.3.1 Las Pestañas de Edición

Las pestañas permiten moverse entre los posibles escenarios de programación, a saber, editar, compilar y revisar y usar la historia.

- **Edición:** Muestra el texto de los programas. Aunque su contenido es editable, hay que tener en cuenta que si se hacen cambios manuales que afecten los principios de refinamientos no se tiene garantía de la corrección del programa final. La figura 1 muestra una vista de la pestaña de edición.

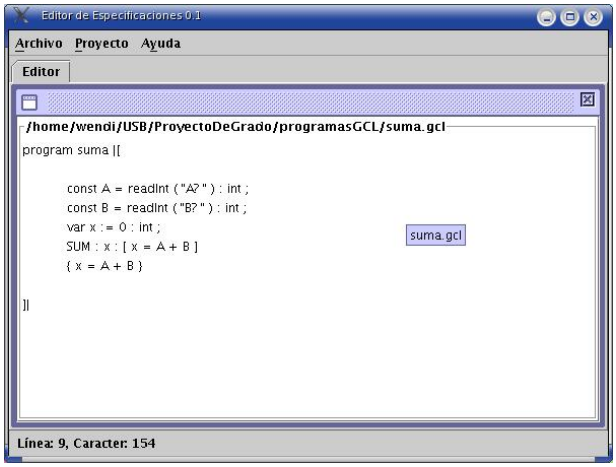


Figura 1: Vista de la pestaña de edición en el editor

- **Compilación:** Muestra el resultado de la compilación de los programas. Esta pestaña aparece cuando se selecciona desde la pestaña de edición la opción compilar y muestra un mensaje indicando que el programa fue compilado exitosamente si el reconocedor del lenguaje L no encontró errores, y en caso contrario despliega los errores.

- **Historia:** Muestra la historia del refinamiento. Aparece cuando se selecciona "Ver Historia". Está formada por dos cuadros de texto separados horizontalmente por una barra; el cuadro superior muestra la historia del refinamiento en sesiones anteriores, y el inferior, en la sesión actual. Ver figura 2
  Cuando se abre el archivo de un programa P que se ha refinado, se cargan en una pila las transiciones de transformaciones que se le han realizado en sesiones anteriores a P. Cuando se realiza una transformación, se guarda en la pila información sobre la transformación que se aplicó. Al terminar la sesión, la historia del refinamiento se guarda en un archivo en el mismo directorio donde se encuentra el archivo del programa P, con el nombre del archivo y extensión ".history". En ese archivo se guarda un árbol XML de la secuencia de transformaciones que quedaron en la pila.

4.3.2 Refinamiento

Reglas aplicables y no aplicables. Cuando se desea aplicar una regla de refinamiento sobre una especificación del programa que se encuentra visible en la pestaña de edición, se necesita que el cursor esté colocado sobre algún caracter de ese componente y llamar a la opción de refinar. Al invocar a refinar se ejecuta el reconocedor de programas y si el programa es aceptado se almacena y se obtiene su representación como

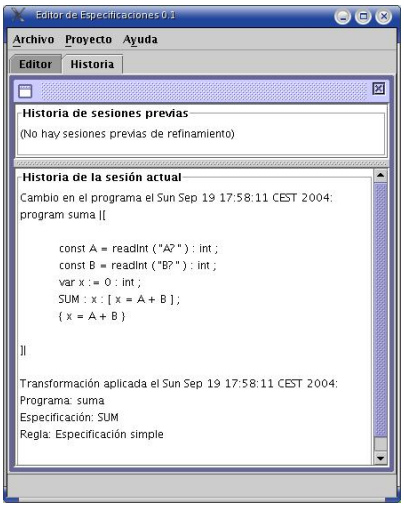


Figura 2: Vista de la pestaña de la historia

objeto. Para identificar el caracter de inicio y de final del componente, se recorre la lista de subprogramas y de acciones del cuerpo hasta encontrar la acción que tiene entre sus límites el caracter donde está colocado el cursor. Usando esta acción, el subprograma donde se encuentra y el bloque donde se encuentran la acción anterior y posterior se determina qué reglas de refinamiento son aplicables a ese componente.

El editor revisa cuáles de las reglas definidas son aplicables a la especificación seleccionada y se aplican las plantillas de las condiciones de aplicabilidad de cada una de esas reglas al árbol XML de esa construcción. Existen tres reglas que son excepcionales, porque no se verifican sobre la especificación sino sobre el bloque o la secuencia de acciones en la que se encuentran:

1. *Absorber suposición*:
   La plantilla de condición de aplicabilidad revisa si en el bloque donde se encuentra el cursor existe alguna especificación precedida de una aserción.

2. *Composición de skip*:
   La plantilla de condición de aplicabilidad verifica si el bloque donde se encuentra el cursor o alguno de sus bloques internos, comienza o termina con una instrucción `skip`.

3. *Remover constante o variable*:
   La condición de aplicabilidad revisa si existe alguna variable o constante declarada en el bloque donde se encuentra el cursor que no esté siendo utilizada.

En cada aplicación de las plantillas de condiciones de aplicabilidad se retorna un resultado que indica si la regla es aplicable o no. Si es aplicable, su identificador se agrega a una lista de las reglas aplicables; en caso contrario, se guarda en una lista de las reglas no aplicables.

Selección de una regla Cuando ya se ha definido cuáles reglas de refinamiento son aplicables y cuáles no. Si no hay reglas aplicables, el editor muestra una ventana de diálogo indicando que ninguna regla es aplicable, y se le permite al usuario ver otra ventana donde se muestra la lista de todas las reglas. Al seleccionar una de estas reglas se puede ver su descripción, los parámetros que necesita y los requerimientos sintácticos de la acción a la que se le puede aplicar. Si hay reglas aplicables, aparece un cuadro de diálogo que las muestra y como antes se permite ver su información. La Figura 3. muestra una de estas ventanas.

Aplicación de las reglas de refinamiento. Cuando se va a aplicar una regla de refinamiento se presentan dos escenarios: se requiere o no que el usuario proporcione algún parámetro. Si se necesita algún parámetro, aparece una ventana de dialogo que le solicita al usuario los parámetros correspondientes, se

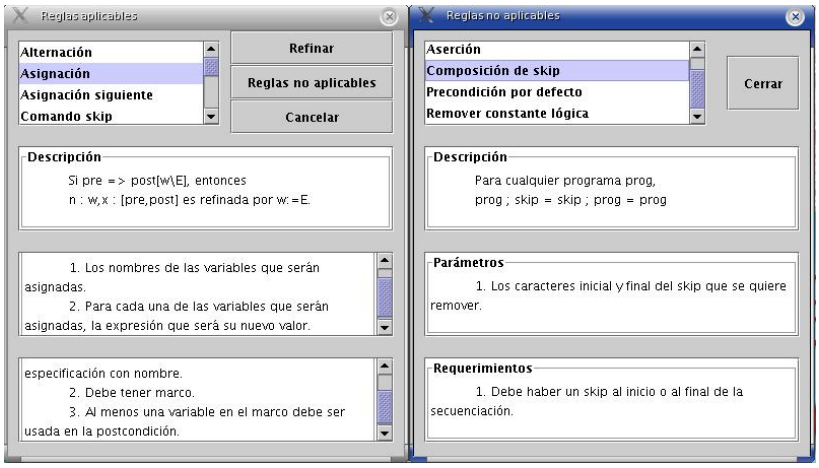


Figura 3: Vista de las ventanas de las reglas aplicables y no aplicables para el programa *suma*

verifica que satisfagan la sintaxis de L, se almacenan y se procede a aplicar la regla, de lo contrario simplemente se aplica la regla. Para aplicar la regla se usa su identificador para cargar la plantilla de la regla y se le pasan los parámetros.

Cuando el motor de XSL transforma el árbol, se escribe en un string todo el programa que se encuentra en la pestaña de edición desde el caracter inicial hasta el caracter anterior al caracter inicial de la construcción que se refinó. Luego ese string se concatena con la representación en texto del árbol de la construcción que se refinó, de acuerdo a la gramática de L, y se concatena finalmente con todo el texto que va desde el caracter que sigue al caracter final de la construcción refinada hasta el final del programa. Esto produce un nuevo programa que se debe desplegar en la ventana del editor.

4.3.3 Navegación por los pasos de refinamiento

Las transformaciones hechas sobre un programa se guardan en una pila, por la cual se puede "navegar", esto es, se pueden deshacer transformaciones realizadas, y en caso de haber deshecho transformaciones, rehacerlas. Se puede navegar por transformaciones que se hicieron en sesiones anteriores. Esto permite que el usuario regrese a un estado anterior del programa y pueda decidir si toma un nuevo camino de transformación, usando reglas diferentes, o parámetros distintos.

Al navegar hacia atrás en la pila de transformaciones no se pierden las transformaciones realizadas, sino sólo se cambia la referencia al estado que se quiere observar en la pestaña de edición. Cuando el usuario deshace transformaciones y aplica una regla de refinamiento al programa o algún componente del programa, las transformaciones que se habían realizado se pierden, quedando en la pila las transformaciones hechas hasta el estado desde donde se aplicó la transformación, y en el tope, una nueva transformación, que es la última aplicada. También se puede navegar hacia adelante en la pila de transformaciones. Esto cambia la referencia del estado actual al estado siguiente, el cual se observa en la pestaña de edición.

5 Observaciones Finales

Se obtuvo un editor que permite transformar una especificación en un programa ejecutable que respeta la especificación original. Como ya se mencionó, todos los programas intermedios están escritos en un subconjunto de *GaCeLa* extendido, no obstante, una vez se prueba que el programa es correcto, el mismo se puede traducir mecánicamente a cualquier lenguaje de alto nivel. En su estado actual el editor implementa un subconjunto de las reglas propuestas por Morgan, deja que el usuario tome las decisiones de diseño y maneja el trabajo sucio de determinar si se cumplen las condiciones de aplicabilidad de las reglas y efectuar las transformaciones. También es capaz de mostrar una documentación sobre las reglas de refinamiento implementadas (definición, parámetros, requerimientos) y permite revisar la historia del proceso de refinamiento. El editor se distribuye bajo licencia GPL.

Esta herramienta puede ser particularmente útil en la enseñanza de métodos formales para la deducción de programas. Creemos que, bien orquestado, podría usarse en cursos introductorios en conjunto con *GaCeLa*, pero seguro que resultaría de mucho provecho en cursos avanzados. Un desarrollador con cierta experiencia en métodos formales podría usarla como una herramienta de asistencia.

Al aplicar una regla, que requiera alguna condición de aplicabilidad, el sistema agrega automáticamente obligaciones de prueba que el programador debe verificar manualmente para poder asegurar la corrección del programa derivado. Este proceso se podría automatizar para liberar al usuario del trabajo de probar manualmente estas obligaciones. Se propone usar un demostrador automatizado de prueba como ACL2 para llevar a cabo esta tarea.

Entre los puntos que se deben tratar en el futuro figuran: 1) Incluir las reglas que tienen que ver con el tratamiento de arreglos y las que involucran llamadas a procedimientos. 2) Proveer un mecanismo que permita automatizar la demostración de las coerciones que se generan en el proceso. 3) Investigar cómo podríamos usar este editor para hacer refinamiento de datos. 4) Estudiar condiciones de *factibilidad* de una especificación [9].

Agradecimientos

Los autores queremos expresar nuestra gratitud al profesor Jesús Ravelo de la Universidad Simón Bolívar por sus valiosos comentarios que permitieron mejorar una primera versión de este artículo y a los revisores de la conferencia por sus oportunas sugerencias.

Referencias

[1] Manfred Broy and Bernd Krieg-Bruckner. Derivation of invariant assertions during program development by transformation. *ACM Trans. Program. Lang. Syst.*, 2(3):321–337, 1980.

[2] R. M. Burstall and J. Darlington. A transformation system for developing recursive programs. *J. ACM*, 24(1):44–67, Jan. 1977.

[3] Edsger W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8):453–457, 1975.

[4] Edsger W. Dijkstra. *A Discipline of Programing*. Prentice-Hall, Englewood Cliffs, 1976.

[5] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[6] Matt Kaufmann and J. Strother Moore. An industrial strength theorem prover for a logic based on common lisp. *IEEE Trans. Software Eng.*, 23(4):203–213, 1997.

[7] Z. Manna and Waldinger. Synthesis: Dreams → programs. *IEEE Trans. Software Engineering*, 4:294–328, 1979.

[8] Carroll Morgan. The specification statement. *ACM Trans. Program. Lang. Syst.*, 10(3):403–419, 1988.

[9] Carroll Morgan. *Programming from specifications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[10] Uday S. Reddy. Transformational derivation of programs using the focus system. In *Software Development Environments (SDE)*, pages 163–172, 1988.

[11] W. L. Scherlis. Program improvement by internal specialization. In *ACM Symp. on Princ. of program. Lang.*, pages 41–49. ACM, 1981.

[12] Ascánder Suárez et al. El wiki de claire. http://gacela.labf.usb.ve/ClaireWiki.

[13] Ascánder Suárez et al. El wiki de gacela. http://gacela.labf.usb.ve/GCLWiki.

[14] Heather Williamson. *Manual de Referencia XML*. Mc-Graw Hill, 2001.

XSCoRe: A Program Comprehension Workbench

Pablo Montes
Politécnico Grancolombiano, Ing. de Sistemas
Bogotá, Colombia
pmontesa@poligran.edu.co

and

Silvia Takahashi
Universidad de los Andes, Ing. de Sistemas y Computación
Bogotá, Colombia
stakahas@uniandes.edu.co

Abstract

Even software that was developed using modern object oriented languages like Java can profit from reengineering. This is *particularly* true for software systems that were not developed following all the guidelines for developing good quality software.
As always, the first step in the reengineering effort is program understanding. This paper presents an approach for aiding in the understanding of Java programs.
We describe an XML representation that can be used to model Java code. This representation is at a higher level of abstraction than an abstract syntax tree because it incorporates constructs from the underlying object oriented paradigm. A novel characteristic is the inclusion of Javadoc comments in the representation of the Java code.
We also describe a workbench that can be used to aid in the understanding of Java programs. This workbench includes the translator from source code to the XML representation of Java code, as well as specific XLS transformations that can be used to extract information. The workbench can be easily extended with new XLS transformations depending on the needs of the software maintenance engineer.

Keywords: Source code representation, XML, Javadoc comments, XSL transformations, program comprehension workbenches.

Resumen

Inclusive el software que due desarrollado usando lenguajes de programación modernos como Java puede beneficiarse de la reingeniería. Esto es particularmente cierto para sistemas de software desarrollados sin seguir todas las técnicas para el desarrollo de software de alta calidad.
Como siempre, el primer paso en un proyecto de reingeniería es la comprensión de programas. Este artículo presenta un enfoque para ayudar en la comprensión de programas Java.
Definimos una representación XML para modelar código Java. Esta represntación está a un nivel de abtracción más alto que el árbol de sintaxis ya que incorpora constructos del modelo de objetos subyacente. Una característica novedosa es la inclusión de comentarios JavaDoc en la representación del código Java.
También describimos un banoc de trabajo que puede ser usado APRA ayudar en la comprensión de programas Java. Este banco de trabajo incluye un traductor del código fuente a la representación XML así como transformaciones XLS específicas que pueden ser usadas para extraer información. El banco de trabajo puede ser extendido fácilmente con nuevas transformaciones dependiendo de las necesidades del mantenedor de software.

Palabras Claves: Representación de código fuente, XML, comentarios Javadoc, transformaciones XSL, bancos de trabajo para la comprensión de programas.

1. INTRODUCTION

Software reengineering emerges as a field of study in response to the large number of legacy systems that have to be maintained effectively. With the advent of object oriented languages, techniques, and methodologies, it was believed that object oriented software would in itself be maintainable and that no additional tools would be required to aid in program comprehension: all the program artifacts would always be up-to-date, and it would always be possible to know what the actual design of the application was. However, object oriented software also suffers from the lack of good software practices, and we are often faced with the problem of reengineering software written in modern programming languages such as Java. As always, the first step in the reengineering effort is program understanding.

This paper summarizes research carried out in Pablo Montes' Master's Thesis [[17]]. The purpose of the project was to design a tool that would ease the process of program comprehension for Java programs. The result was a workbench for reverse engineering Java programs which we called of **XScoRe**. The backbone of this workbench is **`xjava`**, an XML based representation of Java source code. The workbench includes a translator from Java projects to **`xjava`** and other tools used to extract information from the translated code.

The definition of the XML based representation was a pivotal result of the project because it influenced the development of the workbench. By using XML, the tools that make up the workbench can take advantage of existing techniques for managing XML documents. We can obtain different views of the same model which can, in turn, be used to further the understanding of the software system. Though the workbench described in this paper only implements a few views and their corresponding transformations, it can be easily extended to provide information needed to achieve a better understanding of a software system.

This paper is organized as follows: Section 2 briefly describes the motivation for building a reverse engineering workbench and justification for using an XML based representation. Section 3 describes the approach used to solve the problem. Section 4 describes the actual solution that was developed. Section 5 outlines directions for future research, and Section 6 presents some conclusions.

2. BACKGROUND

In this section we list the characteristics that reengineering tools should have. We also state the motivation for using XML.

2.1. The ideal reengineering workbench

Most authors (see [[10], [18], [19], [20], [21], and [22]]) agree on the characteristics that must be offered by an application that aids in the task of software reengineering. These features are summarized below.

1. The tool must be able to extract information from the software artifacts and generate representations that are at a higher level of abstraction. For example, if the artifact that is being studied is the source code, the representation of the corresponding abstract syntax tree would be at a higher level of abstraction; when dealing with a design document, it would be helpful to obtain analysis information. Examples of analysis information include information regarding the real-world data that is being represented in the application data as well as defining functional and non-functional requirements.
2. Once the information has been extracted, the tool must provide means for exploring the extracted data. This exploration activity is, according to many authors, one of the most important tasks in a reengineering project. To provide facilities for exploration, the tool should offer the following features:
   a. Navigation: Software systems are generally nonlinear and can be viewed as a multidimensional network of artifacts. The tool must facilitate navigating this network to aid in the process of software understanding.
   b. Multiple views: Visual metaphors are useful and are commonly used to represent information and ease its comprehension. The tool must offer, not only information obtained from the extraction phases, but also mechanisms for the software engineer to create new views according to his/her needs.
   c. Extensible: Most reverse engineering applications offer a finite set of features. Though the designers of the application may consider these to be sufficient, it is not possible to predict what aspects of a system are important for all users, nor how to represent them. The application, therefore, must offer mechanisms by which users can extend its functionality.
   d. Information exchange: Currently, there is a widespread development of applications that support software development both for reverse and forward engineering. The reengineering application must be able to

interoperate with other reengineering tools and be able to connect with other systems, by using results from or providing information to other components.

2.2. Using XML

XML is a Standard format for data exchange [2]. It seems inevitable to relate the use of XML with the practice of reverse engineering. XML intrinsically provides or rather promises all the characteristics mentioned above. Moreover, recent and current research has resulted in the definition of mechanisms for: processing XML documents (SAX, DOM [[14]]), generating multiple views from XML documents through XLST transformations [[8]]. These views often involve the use of HTM thus giving the option of hyperlink navigation. Services for querying data in XML documents (using XPATH, XQUERY and XQL) can also be provided. Moreover, XML has an inherently hierarchical structure where markers are used to identify data. Since naming data often gives an idea regarding its meaning, XML is often described as a mechanism that can be used to specify data semantics.

Therefore: Why not define a representation of an application using XML? This would offer the possibility of obtaining information at a higher level of abstraction than the source code, provide the extensibility and support information exchange. We are aware that many other approaches have also used XML as a representation language. In the next section, we describe other approaches and compare them to our proposed solution.

3. A First Step

The source code of a program, though unambiguous due to the fact that it adheres to a well defined grammar, is in a flat text format completely lacking an explicit structure. Its structure can be made explicit after syntactic analysis [[20]]. Usually, the next level of abstraction is the abstract syntax tree (AST). This representation is often used by compilers to represent the source code to analyze, optimize and translate it to a lower level language (e.g., byte code, for Java) [[15]]. This representation is not always intuitive and often reflects the language's syntactic peculiarities instead of higher level language building blocks [[1]].

One of the contributions Java has made to the software engineering community is the standardization of documentation source code comments, informally known as *Javadoc* comments. This provides not only a standard, but also encourages software engineers to document their programs by offering the possibility of automatically generating the basic class documentation in an HTML document. It also serves the reverse engineering community by offering more information that can aid in the process of program understanding. When designing a high level representation for Java source code, *Javadoc* documentation should not be ignored. However, many existing high level representations often do.

In this section we outline the design of XScoRe, the reengineering workbench and **`xjava`**, the source code representation used by the workbench. We begin by outlining other approaches to the problem of representation. We also describe the workbench and give the main characteristics of the representation.

3.1. An XML Based Representation

There are various approaches, mostly at the academic level, for representing source code using XML. Each one has its own characteristics as well as advantages and disadvantages. Some of the most significant approaches are the following:

- SrcML [[4], [5], [20]]
- CppML [[15], [20]]
- OOML [[15]]
- JavaML [[1], [15], [20]]

We studied these approaches, to determine their features –both the good and the bad– and define a representation that integrates the best characteristics of existing approaches. Our goal was to define a mechanism that would aid in the task of reverse engineering and program comprehension.

SrcML, for example, uses a generic approach. It can be used to model source code written in any language. To be able to achieve this efficiently, the source code is semi-analyzed (the method's body and other low level characteristics are not represented) and the code structure, including spaces an line-breaks, is kept intact. By using SrcML, only the high level characteristics can be obtained.

CppML, defines an XML based grammar for C++ source code. It represents source code with its structure as well as the analyzed structure of the syntax tree. This representation adheres to the syntactic structure of the program and does not model high level object oriented concepts.

OOML presents a much more generic approach. Similarly to SrcML, it models only de basic concepts of the object oriented paradigm and, therefore, it does not include all the information found in the source code. On the other hand, it represents all the analyzed information in a tree-like XML document, without taking into consideration the original code structure. Consequently, this representation can be generated, not only from source code, but also from a more complex predefined representation of the source code.
Finally, JavaML is the most complete representation found and the one that provides the most complete documentation. As such, it is the basis for this research work. This representation, however, only takes into account the information needed by the compiler to generate the ByteCode. It, therefore, omits comments which are of great importance in a program comprehension workbench. Additionally, the XML representation is based almost exclusively on the Java grammar and, therefore, it does not take into account the semantic information of the Object Oriented Paradigm.
We decided to use XML as the representation language for modeling the Java syntax. In the defined representation, we also model some aspects related to the semantics of the object oriented paradigm. The representation also supports Javadoc comments. As we mentioned above, we called this representation **`xjava`**. The novel aspects of our representation are, as we have said, that it is at a higher level of abstraction than the syntax tree and that we represent comments. In section 4.1, we show an example of the representation of a simple class. It shows how we include information regarding classes and packages that can be used to enhance high level software comprehension.

3.2. Using the representation

Once the code has been translated to **`xjava`**, it can be analyzed, transformed (using XSL transformations), and queried to obtain the required information or to view the parts of the code that are relevant to a particular problem. Some of the views that should be provided by the workbench to aid in the task of program comprehension are listed below.

- Extracting information necessary to construct UML diagrams using XMI.
- Dependency graph and call-tree using GXL to represent these structures.
- Reformatted plain text.
- Hyperlinked text (HTML).
- Documentation view (a la Javadoc).
- Graphical view of the program's structure.
- Querying mechanisms can also be defined using tools such as XPATH.

3.3. Putting it all together

The workbench is made up of the following tools: a translator from source code to **`xjava`** and a few transformations. It is extensible and the initial set of tools and services should not constitute a roadblock to the person in charge of program comprehension. The user can define new transformations and querying mechanisms depending on his/her program comprehension needs. Standard XML processing APIs such as SAX and DOM can be used to process and analyze the extracted data.
Figure 1 illustrates the complete process that is supported by the workbench: all of the source code files are first translated to their equivalent XML representation. Each java file is translated to its corresponding XML file. Additionally, there is a single XML that specifies the packages that make up the system. By using Xlink, we can reference the XML files that represent each individual source code file, indicating to which package it belongs. If we wanted to create a new view of the system we only need to define an XSL transformation and, by running this transformation through the XSLT processor, without modifying the representation or knowing any specifics about the Java grammar, we could create the desired views.

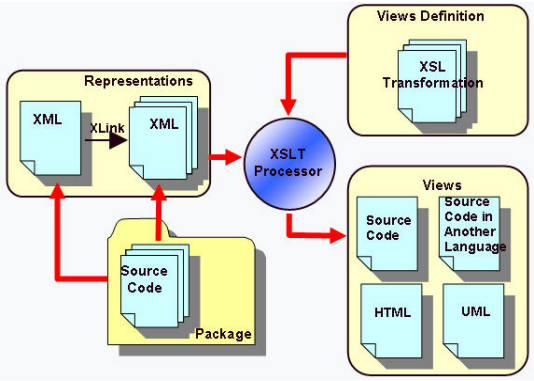


Figure 1. `xjava` and XSL Transformations

The proposed architecture could potentially be used in a software development environment based on various views of a single model. In this way, the problem of maintaining coherence among the source code and all software artifacts could be solved. An environment like this would aid in software maintenance tasks as well as in the complete cycle of software development. This would be particularly helpful in large and complex systems in which many people participate in the process of software development.

The current version of the workbench provides the following transformations: one that generates a documentation view and another for translating the code into an HTML format with hyperlinks. We are currently integrating a tool that obtains the XMI description of the code from **`xjava`** to obtain UML class diagrams that include relations among classes. Each of these examples is not only useful in and of itself but in that it illustrates the fact that the workbench can easily be extended to provide more functionality. This is particularly true in the case of the transformation to XMI, for this was done in a Senior Project by a person who did not participate in the original construction of the workbench.

It is important to point out, that given that **`xjava`** is well defined and documented, anyone who has a good understanding of XML technology can define new views and queries to satisfy new requirements. By adding query services we can provide an environment which is similar to that described in [[1]]. We could even define a new transformation to create a representation that can then be queried by OCL as it is done in [[1]].

4. Implementation Details

This section describes some of the implementation issues. Some **`xjava`** examples for a simple class and a comment are shown. We describe the technical information regarding the actual implementation of the translator and the workbench is provided. Finally we give some examples of how XScoRe has been used.

4.1. The representation: xjava

Once we decided that XML would be the language we would use to define the representation, we proceeded to research the best practices in XML design and manipulation. We followed the guidelines proposed in [[3]].

The main characteristics that the representation should have are the following:

- It should be easily extensible.
- It should be complete: the representation should be able to model any java program that can be compiled by a standard java compiler, and all the details present in the code must be represented. In other words, so that it can be regenerated with a new transformation.
- The representation should be as generic and as simple as possible and should represent the structuring concepts of the object oriented programming paradigm.
- It should also represent *Javadoc* comments: not just as plain text but in a structured manner.

Based on this and on the Java grammar we define **`xjava`**. To achieve extensibility we use an XML schema (XSD) for the definition. We based our design on Java 1.4. We do not include syntactic verification: we assume that the program that is being analyzed does not contain syntax errors. Given that XSD is inherently extensible, it will be easy to incorporate new features that appear in new versions of Java without having to change the representation.

The translator was constructed using publicly available Java grammars. Our representation supports all constructs in this grammar, so its completeness is dependent on the completeness of the published grammars. Note that we are only representing static information. Though it is possible to extract information like the class diagram through transformations, we have not researched how to deal with features such as class reflection.

Since **xjava** is used by other tools that analyze and apply transformations to extract information, it is important that it be generic and simple. Our design reflects this: the generated document's structure is data oriented and is completely structured. We used the Venetian Blind in most of the generated documents.

Some structuring elements are not represented but can be obtained with an adequate transformation. Two examples of these simplifications are: Java's "complete qualified names" for packages, imports, or references to classes. Here, instead of obtaining the name of each subcomponent we always represent the name as a single string. The other example is the case of modifiers for classes, attributes or methods: instead of representing each modifier as a different element, the list of modifiers is represented as a single element which contains a string with all the modifiers separated by blanks. If any of these characteristics is needed, we can define transforms to obtain them.

We show a simple example of a single java file with its corresponding translation to **xjava**. Figure 2 shows the sample code and Figure 3 the corresponding representation.

We also define a representation for a Java project that includes references to all packages and files in a Java application. We used the XLINK standard defined by W3C to manage references and links with XML.

Finally, as we mentioned above, given that the tool is going to be used for reverse engineering it is important to represent comments. We defined a structured representation for comments that follow the Javadoc standard. Other comments are represented in plain text. A problem that arises when analyzing a comment is how to determine what programming construct it refers to. Does it refer to the construct in the same line, the previous line or the following line? At times this depends on the software engineer's or the organization's standards. We could try to guess the corresponding construct through parameterized heuristics depending on the number of blank lines before or after a comment. In our representation, we assumed that the comment made reference to the statement or declaration that appear after it. Figures 4 and 5 show an example of a Javadoc comment and the corresponding representation in **xjava**.

Figures 6, 7, 8 and 9 show the result of applying the XSLT that transforms the **xjava** representation of a more complex program into a series of HTML files that show only the Javadoc comments in an ordered and navigatable manner. This view, though it works like the *Javadoc* tool or Doxygen, is easily modifiable and extensible by including customized tags.

It is important to point out that our tool provides an alternate approach for managing comments. Our tool would be used instead of the *Javadoc* tool. The interesting feature is that comments are analyzed with the code itself. Though our tool would support new types of tags, we do not propose any methodology for writing comments as is the case in [[25]].

```
package co.edu.uniandes.xscore.xjava.test;

public class XJavaTest
{
    private int counter;

    public XJavaTest ()
    {
        counter = 0;
    }

    public int increment (int value)
    {
        counter = counter + value;
    }

    public void print ()
    {
        System.out.println ("Counter: " + counter);
    }
}
```

Figure 2. Sample Program

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xjava:file xmlns:xjava="http://www.uniandes.edu.co/~pmontes/xjava/file"
            name="XJavaTest.java">
  <package namespace="co.edu.uniandes.xscore.xjava.test" />
  <class modifiers="public"
         name="XJavaTest">
    <body>
      <field modifiers="private"
             type="int"
             name="counter"
             dimension="0" />
      <constructor modifier="public">
        <body>
          <assignment operator="=">
            <variable name="counter" />
            <value>
              <literal type="integer"
                       value="0" />
            </value>
          </assignment>
        </body>
      </constructor>
      <method modifiers="public"
              name="increment">
        <returns type="int" />
        <parameter type="int"
                   name="value"
                   dimension="0" />
        <body>
          <assignment operator="=">
            <variable name="counter" />
            <value>
              <addition>
                <variable name="counter" />
                <variable name="value" />
              </addition>
            </value>
          </assignment>
        </body>
      </method>
      <method modifiers="public"
              name="print">
        <body>
          <method_invocation type="method"
                             name="println">
            <target>System.out</target>
            <argument>
              <addition>
                <literal type="string"
                         value="Counter: " />
                <variable name="counter" />
              </addition>
            </argument>
          </method_invocation>
        </body>
      </method>
    </body>
  </class>
</xjava:file>
```

Figure 3. `xjava` representation of sample program

```
/**
 * This is the comment summary. This is the first line of
 * the main description.
 * This is the third line of the main description.
 *
 * @tag_name This is the first line of the tag description.
 *           This is the second line of the tag description.
 * @param parameter_name This is the parameter description.
 * @returns This is the description for the return type and
 *          permissible range of values.
 * @throws
 *   exception_name
 *       This is the first line of the exception description.
 *       This is the second line of the exception description.
 */
```

Figure 4 Javadoc Comment

```
<javadoc>
  <description>
    <line><![CDATA[ This is the comment summary. This is the first line of]]></line>
    <line><![CDATA[ the main description.]]></line>
    <line><![CDATA[ This is the third line of the main description.]]></line>
  </description>
  <tag name="tag_name">
    <description>
      <line><![CDATA[This is the first line of the tag description.]]></line>
      <line><![CDATA[This is the second line of the tag description.]]></line>
    </description>
  </tag>
  <tag name="param">
    <name>parameter_name</name>
    <description>
      <line><![CDATA[This is the parameter description.]]></line>
    </description>
  </tag>
  <tag name="returns">
    <description>
      <line><![CDATA[This is the description for the return type and]]></line>
      <line><![CDATA[permissible range of values.]]></line>
    </description>
  </tag>
  <tag name="throws">
    <name>exception_name</name>
    <description>
      <line><![CDATA[This is the first line of the exception description.]]></line>
      <line><![CDATA[This is the second line of the exception description.]]></line>
    </description>
  </tag>
</javadoc>
```

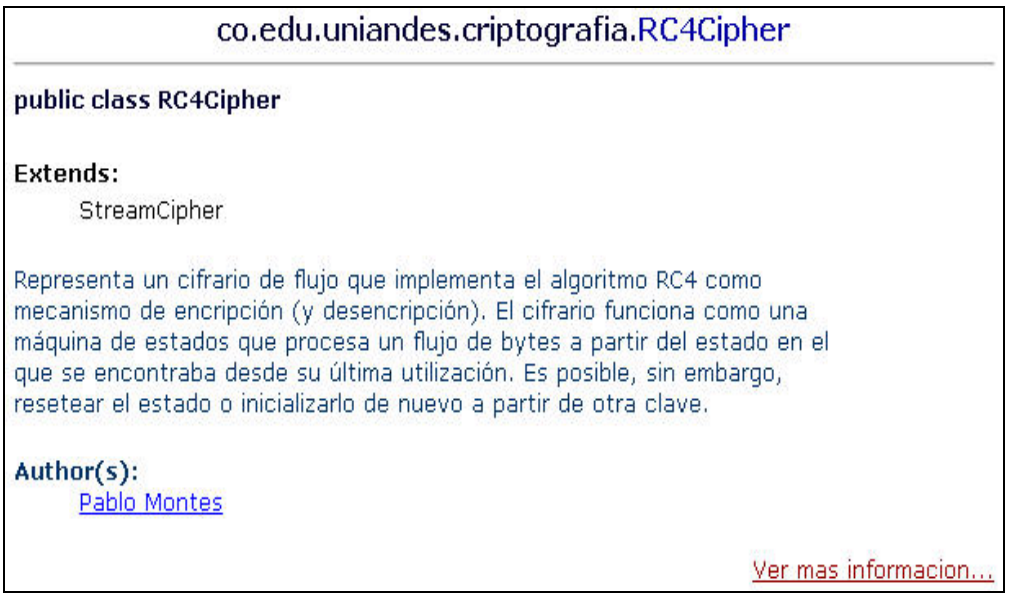
Figure 5. Javadoc Translation

RC4

co.edu.uniandes.criptografia
- Cipher.java
- NotYetInitializedException.java
- RC4Cipher.java
- StreamCipher.java

[default package]
- RC4Test.java

Figure 6 Files list with links



co.edu.uniandes.criptografia.RC4Cipher

public class RC4Cipher

Extends:

StreamCipher

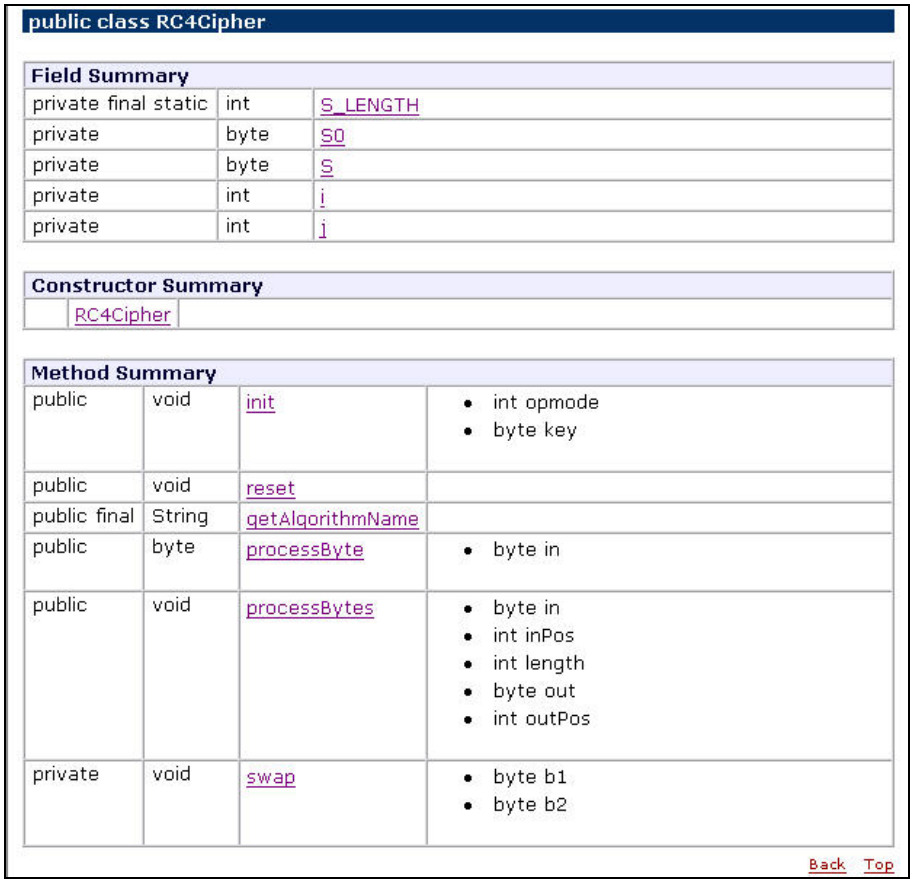
Representa un cifrario de flujo que implementa el algoritmo RC4 como mecanismo de encripción (y desencripción). El cifrario funciona como una máquina de estados que procesa un flujo de bytes a partir del estado en el que se encontraba desde su última utilización. Es posible, sin embargo, resetear el estado o inicializarlo de nuevo a partir de otra clave.

Author(s):

Pablo Montes

Ver mas informacion...

Figure 7 Class summary



public class RC4Cipher

Field Summary		
private final static	int	S_LENGTH
private	byte	S0
private	byte	S
private	int	i
private	int	j

Constructor Summary	
	RC4Cipher

Method Summary			
public	void	init	• int opmode • byte key
public	void	reset	
public final	String	getAlgorithmName	
public	byte	processByte	• byte in
public	void	processBytes	• byte in • int inPos • int length • byte out • int outPos
private	void	swap	• byte b1 • byte b2

Back Top

Figure 8. Class fields, constructors and methods list with links

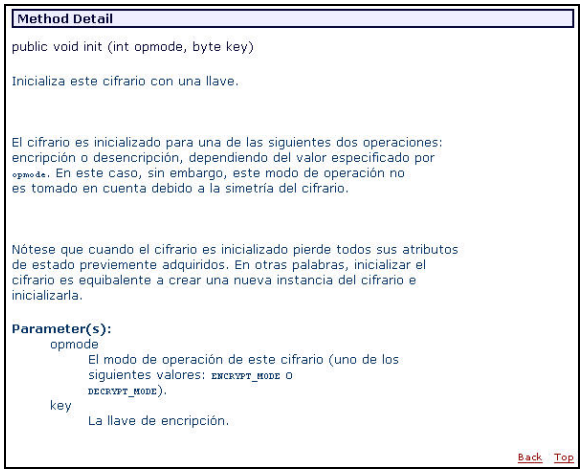


Figure 9. Method detail

4.2. Technical Information

The lexer and parser were generated using JavaCC [[11], [24]]. Currently many developers are using JavaCC and JavaCC grammar files for Java 1.4.2 can be found in the JavaCC Grammar Repository [[12]]. This grammar is based on the initial grammar provided by Sun. We also used JavaCC to analyze Javadoc comments by using only the lexer with lexical states. It is important to point out that though there is a lot of documentation regarding the correct use of Javadoc [[13], [9]], we could not find a grammar that served our purposes.

We first generated the syntax tree using JTB (Java Tree Builder) [[23]], a tool that can be used with JavaCC for this purpose. Then, we used the visitor pattern to generate the corresponding representation in **`xjava`**. We then used JDOM [[14]] to generate the XML document.

Once we obtained this document, transformations can be applied to it to obtain different views. This can all be done though the command line. However, this does not provide a very friendly interface. Therefore, we decided to integrate it with Eclipse's graphical interface through contextual menus. This way a user can translate a Java project to its equivalent **`xjava`** representation and can obtain different views from an **`xjava`** file. Eclipse Wizards must be implemented for each transformation that requires additional user input.

To take advantage of Eclipse's architecture it is not enough to define new plug-ins with new functions. The real potential of the workbench is the fact that it gives the opportunity to extend the plug-ins we have developed. Our plug-in allows developers to add new transformations through XSL transformations or using APIs for XML processing.

4.3. Using XScoRe

The set of tools that we developed were tested on the same code used in their development. This is a large project and we were able to create the **`xjava`** presentation and obtain information given by the XST transformations that we implemented.

The representation of comments was used to create the implementation manual. It is important to point out that by obtaining a representation in XML, it can be visualized in many ways depending on the transformation we use.

We also tested it on various student projects. In particular we used it to create a representation of an application used in a course project in which students have to implement an interface of an existing application. As a rule, students cannot modify the core of the application; they can only use the public methods of the main class. XScoRe aided in the understanding of the existing software.

5. SUMMARY AND FUTURE RESEARCH

In this section we summarize the current state of the XSCoRe workbench and give directions for future work and research. The design of this workbench makes it ideal for adding features because of its extensible architecture.
XScoRe includes the following tools: a translator from Java code to **`xjava`**, and an integrated mechanism for applying XSL transformations on the generated documents. Two views were implemented: a documentation view and a hypertext code view. The first one presents comments similarly to Javadoc's presentation scheme. The second view presents the code using HTML which allows straight forward navigation of the code. XScoRe is currently being enhanced with a transformation that obtained XMI from **`xjava`**. Once we have obtained the XMI representation, UML diagrams can be generated.
The designed system can run as a standalone application through the command line. However, its functionality can be better exploited through the integration with the Eclipse platform.
New transformations can easily be defined and integrated. The representation itself can be used to extract information through queries XPATH, XQUERY or XQL which allow queries a là SQL over XML documents. Currently we are working on constructing queries that can be used to obtain information related to certain metrics that in turn can help us detect bad smells [[6]]. We hope to be able to define transformations that can be used for refactoring.
Not only can the workbench be extended by adding transformations and query possibilities, its design makes easy to incorporate changes to the language's grammar which are quite common with Java's versioning.
In the previous paragraphs we mentioned some directions for future work which are facilitated by the design of the workbench. The basic idea is to include new transformations and support for query management that will add services to the workbench so that it covers more of the activities related to program comprehension and reverse engineering.
In terms of representation of comments, the current representation does not attempt to determine what program element the comment refers to. This would require the used of heuristics and probably some degree of human intervention.
Evidently, it is not enough to have a static representation to achieve program understanding that is enough to carryout maintenance tasks and eventually migrate it to an evolutionary model. Future research should try to define new frameworks in which dynamic information can be represented, preferably also using XML, so that the resulting tools can be integrated with the current system.

6. CONCLUSIONS

In this paper we described the results of a research project whose objectives were to develop tools to ease comprehension of java programs. The main results of the research were the development of a workbench (called XScoRe) and of the underlining representation of Java code (**`xjava`**).
The workbench exhibits many of the characteristics which have been mentioned as vital in a reverse engineering environment: it provides mechanisms for information extraction, generation of multiple views, navigation, and most importantly it provides means for extending it with new features.
The code representation is based on XML. Our approach differs from previous ones, in that the representation used was specifically designed with the purpose of reverse engineering. Therefore, it offers many advantages with respect to the code itself or classical textual code representation: it offers services for other reverse engineering tools to extract information and reason about the existing software system. The proposed representation facilitates obtaining program information in a more accessible and heterogeneous manner. Our representation models structures inherent to the object oriented paradigm instead of only representing Java's grammar rules. Therefore, it is at a higher level of abstraction than the source code and even than the syntax tree. We also represent comments which have been written following the Javadoc standard. It is not common for reverse engineering tools to provide support for managing comments. The representation of comments can also be extended if the user wants to define new tags.
Both XScore and **`xjava`** suggest a new approach towards program understanding. The methodology we used during our research defines a framework for the definition of program understanding workbenches that can be used

to automate this process. The basic idea is to first define a representation of the source code that can be easily manipulated. The defined representation can then be used to construct different views directly from the representation which contains more information than what is usually available in the abstract syntax tree.
Novel features of XSCoRe are the use of XML schemas which makes the workbench more extensible than other approaches; the inclusion of Javadoc comments in the representation so that they can be analyzed automatically; and the integrated representation of all files in an application organized in packages.
The use of XML in a context different from that of data representation and exchange proposes a new approach that can be applied to reverse engineering and to any task that requires representing source code at a higher level of abstraction. This approach can be used in formal verification of programs, automatic code generation, or translation from a programming language to another. By the same token, it can also be used in forward engineering by aiding in the whole process of software development.
We believe that this research is a significant contribution to problem of understanding source code. Moreover, this research gives new perspectives to the use of XML for a purpose different than that of data representation and exchange in multi-tier environments. Clearly the use of XML for program comprehension and development is just beginning. The results we have obtained till now may serve as a stepping stone for building more complex software comprehension tools.

References

[1] Antoniol, G. and Di Penta, M. and Merlo, E. YAAB (yet another AST browser): using OCL to navigate ASTs the 11th IEEE International Workshop on Program Comprehension, Portland, Oregon, USA, May 10-11 2003, pages:13-22; IEEE Computer Society Press, 2003.

[2] Badros, G., JavaML : a markup language for Java source code [online]. International Conference On The World Wide Web (9° : 2000 : Amsterdam, Neatherlands). Available from citeseer: <http://citeseer.ist.psu.edu/badros00javaml.html>.

[3] Bonneau, S., et al. XML Design Handbook. Wrox Press, 2003.

[4] Collard, M., Maletic, J., and Marcus A., Source code files as structured documents. En : 10th International Workshop On Program Comprehension, Paris, France, 2002). [online] Available from citeseer: <http://citeseer.ist.psu.edu/maletic02source.html>.

[5] Collard, M.; Maletic, J., And Marcus A.,. Supporting Document and Data Views of Source Code [online]. Available from citeseer: <http://citeseer.ist.psu.edu/collard02supporting.html>.

[6] Fowler, M., et. al., *Refactoring: Improving the Design of Existing Code*, Addison-Wesley Pub Co, 28 June, 1999.

[7] Friendly, L. The Design of Distributed Hyperlinked Programming Documentation, International Workshop On Hypermedia Design, Montpellier, France, 1995. [online] Available from citeseer: <http://citeseer.ist.psu.edu/friendly95design.html>.

[8] Gardner, J., Rendon, Z., XSLT and XPATH: A Guide to XML Transformations. Prentice Hall, 2001.

[9] How to Write Doc Comments for the Javadoc tool [online], Available from JavaSun <http://java.sun.com/j2se/javadoc/writingdoccomments/index.html>.

[10] JAHNKE, Jen et al. Reverse Engineering: A Roadmap Future of Software Engineering, Limerick, Ireland. [online], Available from citeseer: <http://citeseer.ist.psu.edu/muller00reverse.html>.

[11] JavaCC Documentation [online], Available from JavaSun <https://javacc.dev.java.net/doc/docindex.html>.

[12] JavaCC Grammar Repository [online], Available from JavaSun: <http://www.cobase.cs.ucla.edu/pub/javacc/>.

[13] Javadoc Home Page [online], Available from JavaSun: <http://java.sun.com/j2se/javadoc/>.

[14] JDOM Home Page [online]. <http://www.jdom.org/>.

[15] Kontogiannis, K., Mamas, E. Towards Portable Source Code Representations Using XML, 7th Working Conference On Reverse Engineering, Brisbane, Australia 2000. Available from csdl.computer.org. <http://csdl.computer.org/comp/proceedings/wcre/2000/0881/00/08810172abs.htm>.

[16] Kontogiannis, K., Zou, Y., Towards A Portable XML-based Source Code Representation, Xml Technologies And Software Engineering, 2001. [online], Available from citeseer: <http://citeseer.ist.psu.edu/500234.html>.

[17] Montes, P., XSCoRe: Representación de Código Fuente basada en XML como una Herramienta de Asistencia al Proceso de Ingeniería en Reversa, Tesis de Maestría, Dept. Ingeniería de Sistemas y Computación, Universidad de los Andes, Bogotá, Colombia, 2004.

[18] Rada, Roy. Reengineering Software: How to Reuse Programming to Build New, State-Of-The-Art Software. Fitzroy Dearborn Publishers, 1999.

[19] Santanu, P., Smith, D., and Tilley, S., Towards a Framework for Program Understanding, 4th Workshop On Program Comprehension, 1996. [online], Available from citeseer: <http://citeseer.ist.psu.edu/266671.html>.

[20] Simic, H., Topolnik, M., Prospects Of Encoding Java Source Code In XML, 7th International Conference On Telecommunications, Zagreb, Croatia, 2003 [Online], Available From Citeseer: <http://citeseer.ist.psu.edu/simic03prospects.html>.

[21] Tilley, S., Perspectives on Legacy System Reengineering Reengineering centre, Software Engineering Institute, Carnegie Mellon University, 1995, [online]. Available from www.sei.edu: <http://www.sei.cmu.edu/reengineering/lsysree.html>.
[22] Tilley, S., A Reverse-Engineering Environment Framework [online]. Reengineering centre, Software Engineering Institute, Carnegie Mellon University, 1998. [online]. Available from www.sei.edu: <http://www.sei.cmu.edu/publications/documents/98.reports/98tr005/98tr005abstract.html>.
[23] The Java Tree Builder Homepage [online]: <http://www.cs.purdue.edu/jtb/index.html>.
[24] The JavaCC FAQ [on line] <http://www.engr.mun.ca/~theo/JavaCC-FAQ/>.
[25] Torchiano, M.,Documenting Pattern Use in Java Programs, In Proc. IEEE Int. Conf. on Software Maintenance (ICSM 2002), Montreal, Canada, October 3-6, 2002, pp. 230-233.

Dos Requisitos à Qualificação de Componentes: uma Abordagem Baseada em Refinamentos Sucessivos

Fernando Vieira Paulovich
Wanderley Lopes de Souza
Célio Estevan Moron

Universidade Federal de São Carlos
Departamento de Ciências da Computação
São Carlos-SP, Brazil, Caixa Postal 676
{paulovic,desouza,celio}@dc.ufscar.br

Abstract

This paper presents an approach for the component-based software development, starting with functional requirements modelling and finishing with a pre-selection of components which are candidates for reuse. This approach combines a formal description technique, called *Architectural Modelling Box for Enterprise Redesign (AMBER)*, with *Use Cases Diagrams* on a process which allow us to assure that the selected components preserve what was defined as system functional requirement.

Keywords: Component-based development, Software development, Formal methods.

Resumo

Este artigo apresenta uma abordagem para o desenvolvimento de software baseado em componentes que parte da modelagem dos requisitos funcionais e termina em uma pré-seleção dos possíveis componentes candidatos ao reuso. A mesma combina uma técnica de descrição formal, conhecida como *Architectural Modelling Box for Enterprise Redesign (AMBER)*, com *Diagramas de Casos de Uso* em um processo que permite assegurar que os componentes selecionados preservam o que foi definido como requisito funcional para o sistema.

Palavras-chave: Desenvolvimento baseado em componentes, Desenvolvimento de software, Métodos formais.

1 INTRODUÇÃO

O crescente aumento da complexidade dos sistemas de software vem acarretando em um alto custo de desenvolvimento, em uma baixa produtividade e em um difícil gerenciamento da sua qualidade, fomentando assim a demanda e a busca por novas e eficientes abordagens para seu desenvolvimento.

Umas das mais promissoras abordagens apresentadas até agora, conhecida como *Desenvolvimento Baseado em Componentes (DBC)*, está centrada na idéia de que sistemas de software podem ser desenvolvidos através da seleção e montagem de pedaços reutilizáveis de software [14]. Um pedaço de software, também denominado componente, é um bloco binário de construção auto-contido que provê um único serviço, o qual pode ser usado individualmente ou em composição com outros componentes para o fornecimento de serviços mais complexos [19].

Embora o DBC possa reduzir significativamente o custo e o tempo de desenvolvimento de um sistema de software [15], este ainda carece de maturidade a fim de que possa ser aplicado em escala industrial. Dentre os principais problemas apresentados destaca-se a ausência de mecanismos consistentes que permitam determinar quais componentes pré-existentes implementam satisfatoriamente as funcionalidades propostas para o sistema em desenvolvimento.

Neste sentido este artigo apresenta uma abordagem baseada em refinamentos sucessivos que busca enriquecer o DBC na medida em que combina o modelo arquitetônico formal *Architectural Modelling Box for*

Enterprise Redesign (AMBER) [17, 13] com diagramas de casos de uso da *Unified Modeling Language (UML)* [9, 18] em um processo que parte da definição dos requisitos funcionais do sistema, passa pela definição de sua arquitetura e termina na pré-seleção de componentes candidatos ao reuso.

O restante deste artigo está estruturado da seguinte forma: a seção 2 apresenta uma revisão de algumas abordagens de DBC já existentes; a seção 3 fornece uma visão geral da abordagem proposta; a seção 4 mostra as principais características do AMBER; a seção 5 apresenta a modelagem dos requisitos funcionais do sistema; a seção 6 descreve a transformação da modelagem de requisitos na modelagem da arquitetura do sistema; a seção 7 propõe um processo de seleção de componentes candidatos ao reuso; finalizando, a seção 8 tece algumas conclusões relativas a este trabalho e propõe idéias para trabalhos futuros.

2 TRABALHOS RELACIONADOS

A princípio, o processo do DBC pode ser comparado à engenharia de software convencional ou orientada a objetos. Tal processo tem início no estabelecimento dos requisitos para o sistema a ser construído, seguindo com o projeto arquitetônico do mesmo. Porém, ao invés de prosseguir com tarefas mais detalhadas de projeto, há uma análise do que foi especificado como requisito de forma a buscar e selecionar os componentes disponíveis que possam implementar os requisitos definidos [16].

Na atualidade, várias abordagens para o DBC vêm sendo propostas. Uma das mais citadas, a *Catalysis* [4], na verdade não se trata de um processo de desenvolvimento, mas sim de um modelo de processo que pode ser adaptado de acordo com um projeto em particular. Apesar da Catalysis ser amplamente difundida, a mesma não fornece explicitamente mecanismos para a classificação, busca e seleção de componentes, apenas sugere que esses mecanismos devam existir. Normalmente, tais mecanismos são baseados nas características das interfaces dos componentes [16]. Porém, esse tipo de classificação não consegue fornecer informações suficientes de forma a ser possível estabelecer um mecanismo eficiente de busca e seleção.

Uma maneira de tratar esse problema é utilizar, além da descrição das interfaces dos componentes, especificações mais detalhadas sobre suas funcionalidades. De forma a possibilitar a automatização dos mecanismos de busca e seleção de componentes, especificações formais podem ser empregadas nessa descrição. Tais tipos de especificações (normalmente) possibilitam tal automação por apresentarem uma base axiomática que permite determinar se duas especificações são equivalentes - a descrição de um determinado componente e uma descrição de busca. Nesse sentido, trabalhos foram propostos [7, 8, 12], porém os mesmos não tratam de como os modelos gerados para os requisitos podem ser utilizados como base para tais buscas, impossibilitando verificar se os requisitos propostos para o sistema são implementados pelos componentes escolhidos.

Por outro lado, outras abordagens, tais como [5, 6], tratam em como mapear os requisitos em componentes, mas falham em não prover mecanismos que possibilitem uma verificação se os requisitos propostos são satisfatoriamente implementados nos componentes reusados/criados, ou mesmo quais requisitos estão implementados nesses componentes.

É nesse cenário que a abordagem aqui proposta foi concebida: a de possibilitar que a partir da modelagem dos requisitos do sistema, possíveis componentes candidatos ao reuso sejam escolhidos, garantindo que esses requisitos, ou pelo menos parte deles, estejam implementados nos componentes escolhidos. Na próxima seção, o processo do DBC será detalhado e uma visão geral da abordagem aqui proposta será apresentada, posicionando a mesma dentro desse processo.

3 VISÃO GERAL DA ABORDAGEM

O processo DBC, ilustrado na Figura 1, tem início na fase de Análise, segue com a realização de um Projeto Arquitetônico e termina com um grupo de atividades que devem ser realizadas de forma a povoar, por meio de componentes que estejam disponíveis e/ou sejam criados, a arquitetura definida [3].

No caso da criação de novos componentes a Engenharia de Componentes deve ser empregada, enquanto que no caso da reutilização, a Qualificação e a Adaptação de Componentes devem ser executadas. Na Qualificação, componentes pré-existentes são selecionados tomando como base a arquitetura modelada para o sistema. Na Adaptação os componentes selecionados são adaptados às necessidades dessa arquitetura. Finalmente, na Composição os componentes criados/selecionados são integrados de maneira a compor o sistema, o qual deve ser testado e, quando necessário, ter seus componentes atualizados.

A abordagem proposta na Figura 2 insere-se no processo DBC apresentado na Figura 1. Nessa abordagem, a Modelagem dos Requisitos, que faz parte da fase de Análise, a Modelagem da Arquitetura, que

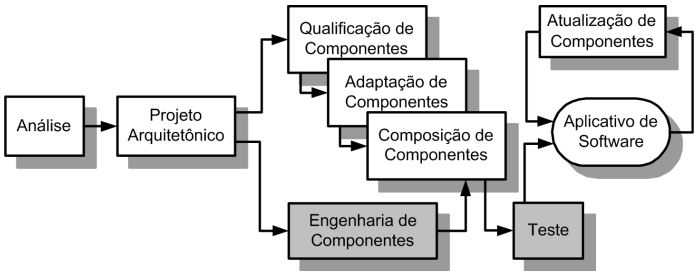


Figura 1: Processo DBC.

faz parte do Projeto Arquitetônico, e a Qualificação de Componentes definem um sub-processo no processo DBC que objetiva selecionar os componentes candidatos ao reuso, assegurando que os mesmos preservam a funcionalidade inicialmente proposta como requisito para o sistema.

Esse sub-processo pode ser dividido em duas etapas distintas. Na primeira etapa os requisitos são modelados, delimitando-se o sistema e definindo-se a sua funcionalidade. Uma vez realizada tal modelagem, os requisitos são transformados, por meio da aplicação de regras bem definidas, na arquitetura inicial do sistema (1). Na segunda etapa os componentes são qualificados e a arquitetura é refinada, sendo essa etapa executada sucessivamente até que todos os componentes, que possam compor o sistema, tenham sido identificados.

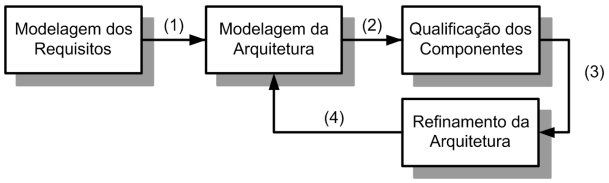


Figura 2: Abordagem proposta.

Na qualificação de componentes os modelos referentes às partes da arquitetura são comparados aos modelos referentes aos componentes pré-existentes a fim de verificar se existem funcionalidades equivalentes (2). Nos casos afirmativos, tais componentes se tornam possíveis candidatos ao reuso. Se após essa qualificação ainda restarem partes definidas na arquitetura que não foram assinaladas a componentes disponíveis (3), os modelos referentes a essas partes são refinados e a qualificação é novamente aplicada aos modelos obtidos nesse refinamento (4).

Esse processo de refinamento e qualificação é repetido até que todas as partes definidas na arquitetura do sistema tenham sido assinaladas a algum componente. No caso da arquitetura ter atingido o seu nível máximo de detalhamento (isso é uma medida subjetiva) e ainda restarem partes que não tenham sido assinaladas a componentes disponíveis, novos componentes devem ser desenvolvidos de forma a implementar tais partes, sendo que os modelos referentes a essas partes se tornam as especificações para a classificação desses novos componentes.

4 ARCHITECTURAL MODELLING BOX FOR ENTERPRISE REDESIGN (AMBER)

Partindo-se da análise das deficiências da técnica de descrição formal *Language of Temporal Ordering Specification (LOTOS)* [22], um grupo de pesquisadores da Universidade de Twente, Holanda, desenvolveu um modelo arquitetônico buscando resolver alguns dos problemas encontrados. Tal modelo, conhecido como AMBER, possibilita a especificação de sistemas em termos de estruturas de entidades funcionais com seus respectivos comportamentos.

4.1 Entidades Funcionais

Um sistema pode ser modelado em termos de uma estrutura de entidades funcionais que interagem entre si, cada entidade representando uma parte lógica ou física do mesmo. De forma a indicar que duas ou mais entidades interagem entre si, pontos de interação devem ser usados.

A Figura 3 representa a modelagem de uma rede de comunicação e seus usuários *A* e *B*. Graficamente, uma entidade funcional é representada por meio de um retângulo com as bordas cortadas, enquanto que um ponto de interação é representado por uma elipse.

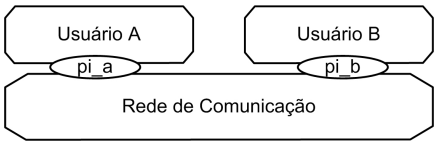


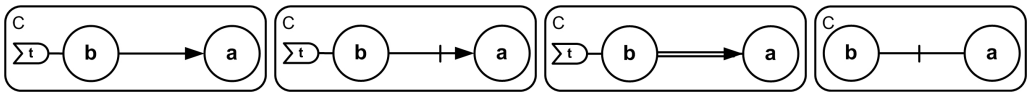
Figura 3: Entidades representando uma rede de comunicação e seus usuários.

4.2 Comportamentos

A cada entidade funcional existe um comportamento associado, o qual representa sua funcionalidade. Um comportamento é definido como uma coleção de atividades que a entidade pode executar isoladamente ou em cooperação com outras entidades. De forma a se modelar uma atividade executada por uma única entidade, o conceito de ação é usado. Caso uma atividade deva ser executada por duas ou mais entidades em cooperação, usa-se o conceito de interação.

Além da modelagem das atividades que uma entidade pode executar, o relacionamento entre tais atividades deve ser representado. De modo geral, esse relacionamento define as condições para a ocorrência de uma (inter)ação em termos da ocorrência ou não de outras (inter)ações. Para a modelagem dessas condições, o conceito de relação de causalidade é empregado.

Existem três condições básicas que podem ser modeladas em uma relação de causalidade. Supondo que `a` e `b` sejam ações de um comportamento, podemos modelar a relação de causalidade de `a` em relação a ocorrência ou não de `b` como: (1) a ocorrência de `b` é uma condição para a ocorrência de `a`; (2) a não ocorrência de `b` é uma condição para a ocorrência de `a`; e (3) a ocorrência de `b` é uma condição para a ocorrência de `a`, tal que `b` deve ocorrer simultaneamente com `a`. A Figura 4 apresenta a modelagem dessas três relações básicas. Graficamente uma ação é representada por um círculo e um comportamento por um retângulo de cantos arredondados.



(a) a ocorrência de b é requisito para a ocorrência de a.

(b) a não ocorrência de b é requisito para a ocorrência de a.

(c) b deve ocorre simultaneamente a a.

(d) escolha entre a ou b.

Figura 4: Representação gráfica das condições básicas de causalidade.

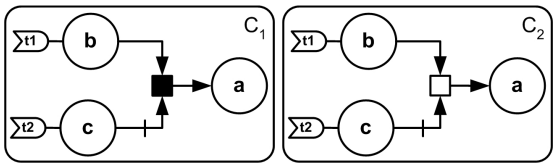
Por outro lado, podem existir ações que não dependam de outras, podendo ocorrer, sem restrições, a partir do início da execução de um comportamento. Tais ações são chamadas de ações iniciais e são representadas como ações habilitadas apenas por um mecanismo chamado *trigger*. Na Figura 4 (a), (b) e (c), todas as ações `b` são ações iniciais.

Além dos relacionamentos básicos entre (inter)ações, AMBER oferece outros que são composições entre esses. Um dos mais importantes, o *Relacionamento de Escolha*, é equivalente a dois relacionamentos de desabilitação, onde a ocorrência de uma (inter)ação desabilita a ocorrência de outra e vice-versa. Esse relacionamento, apresentado na Figura 4 (d), é representado graficamente por uma linha ligando duas (inter)ações com uma barra cortando a mesma. Para maiores informações consulte [17].

Em alguns comportamentos, múltiplas condições básicas envolvendo diferentes ações devem ser satisfeitas para que uma determinada ação possa ocorrer. Esta situação pode ser representada através da conjunção de

duas ou mais condições básicas envolvendo diferentes ações. Para esta modelagem, o operador de conjunção deve ser usado. Por exemplo, se a, b e c são ações de um comportamento, podemos modelar por meio do operador de conjunção que a ocorrência de b e a não ocorrência de c são condições para a ocorrência de a. Na Figura 5 (a) esta conjunção de condições é modelada. Graficamente o operador de conjunção é representado por um quadrado preenchido.

Por outro lado, existem comportamentos em que pelo menos uma condição básica deve ser satisfeita para que uma determinada ação possa ocorrer. Esta situação pode ser representada através da disjunção de duas ou mais condições básicas de causalidade. Para esta modelagem, o operador de disjunção deve ser usado. Por exemplo, se a, b e c são ações de um comportamento, podemos modelar por meio do operador de disjunção que a ocorrência de b ou a não ocorrência de c são condições mínimas e suficientes para a ocorrência de a. Na Figura 5 (b) essa disjunção de condições é modelada. Graficamente o operador de disjunção é representado por um quadrado não preenchido.



(a) a deve e b não deve ocorrer antes de c.

(b) a deve ou b não deve ocorrer antes de c.

Figura 5: Conjunção e disjunção de condições básicas de causalidade.

5 MODELAGEM DOS REQUISITOS DO SISTEMA

Para a modelagem dos requisitos funcionais será usado o modelo de comportamento AMBER em combinação com diagramas de casos de uso. Os diagramas de caso de uso representam atores, que modelam entidades externas que interagem com o sistema (ambiente do sistema), e os chamados casos de uso, que modelam a funcionalidade provida a tais entidades como operações suportadas pelo mesmo [18].

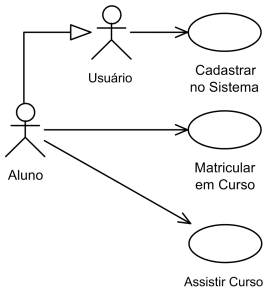
Apesar de qualquer nível de granularidade ser permitido na definição dos casos de uso, não havendo métricas estabelecidas para determinar um correto nível para o projeto de um software, fica definido que esse deve permanecer em um alto nível de abstração, onde cada caso de uso representa um aspecto funcional ortogonal da aplicação sendo desenvolvida [18].

De forma a completar e tornar mais precisa a descrição da funcionalidade provida pelo sistema, comportamentos devem ser especificados para cada caso de uso. Esses comportamentos, conhecidos como cenários, consistem de conjuntos de seqüências de ações que o sistema desempenha para produzir um resultado esperado por seus usuários. Para que tais ações estejam de acordo com o nível de abstração requerido para essa fase de desenvolvimento, tais seqüências devem ser especificadas a partir do ponto de vista do ambiente do sistema, definindo as interações que podem ocorrer entre ambiente e sistema e os relacionamentos entre tais interações.

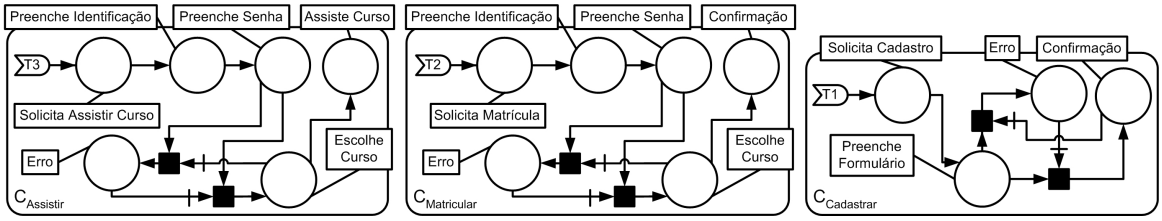
Na atualidade, as técnicas mais empregadas nessa especificação utilizam linguagem natural (texto), pseudo-código ou máquinas de estado. Na abordagem proposta será utilizado o modelo de comportamento AMBER. Assim, além das características que o tornam mais preciso e representativo que as técnicas vigentes, como alto nível de abstração e semântica formal, essa especificação servirá como ponto inicial para a definição da arquitetura do sistema, onde AMBER será aplicado.

Apesar de estarmos modelando interações, o conceito AMBER utilizado é o conceito de ação. Isto se dá, pois, nessa fase de desenvolvimento não há uma divisão de responsabilidades e restrições entre sistema e seu ambiente na execução dessas interações, resultando em modelos que representam o serviço (total) oferecido pelo sistema.

A Figura 6 (a) apresenta modelagem de um diagrama de casos de uso para um sistema (simplificado) de ensino à distância - consultar [11] para um estudo de caso mais completo. Nessa modelagem são apresentadas três funções principais que permitem ao usuário comum se cadastrar no sistema, tornando-se um aluno do mesmo, e ao aluno se matricular e assistir a um determinado curso. A Figura 6 (b) apresenta os comportamentos correspondentes dos casos de uso identificados.



(a) diagrama de casos de uso.



(b) comportamentos dos casos de uso.

Figura 6: Diagrama de casos de uso e respectivos comportamentos.

Embora cada caso de uso seja independente, sua descrição pode ser fatorada, formando uma composição de casos de uso menos complexos. Nessa fatoração, partes comuns existentes entre casos de uso podem ser assinaladas a novos casos de uso de forma que suas descrições possam ser (re)usadas.

Para representar tal reuso, três tipos de relacionamentos podem existir entre casos de uso, que são: *inclusão*, *extensão* e *generalização*. Na abordagem proposta, somente será considerada a inclusão de um caso de uso em outros e sua repercussão no que tange os comportamentos a eles relacionados.

Na inclusão, as partes comuns dos comportamentos dos casos de uso são assinaladas a um novo comportamento, tornando-se o mesmo o cenário de um novo caso de uso. Além disso, os comportamentos dos casos de uso que contém essa parte em comum devem instanciar no lugar das mesmas esse novo comportamento. Esta instanciação é explicitamente indicada através do conceito AMBER de pontos de entrada/saída (representados pelo símbolo $\triangleright$).

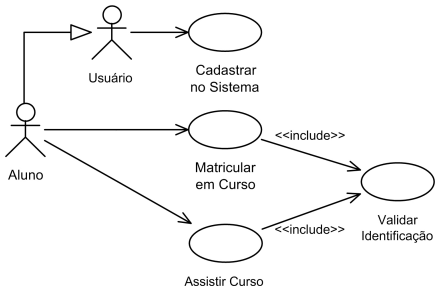
A Figura 7 apresenta a fatoração do que foi modelado na Figura 6. Na Figura 7 (b) as partes comuns dos comportamentos $C_{Assistir}$ e $C_{Matricular}$, modelados na Figura 6 (b), são assinaladas ao comportamento $C_{Validar}$, que por sua vez é explicitamente instanciado dentro dos comportamentos $C_{Assistir}$ e $C_{Matricular}$. Na Figura 7 (a) é mostrada a fatoração do diagrama de casos de uso modelado na Figura 6 (a). Nessa figura os casos de uso *Matricular em Curso* e *Assistir Curso* incluem o caso de uso *Validar Identificação* (inclusão essa indicada por meio do estereótipo $<< include >>$).

6 MODELAGEM DA ARQUITETURA DO SISTEMA

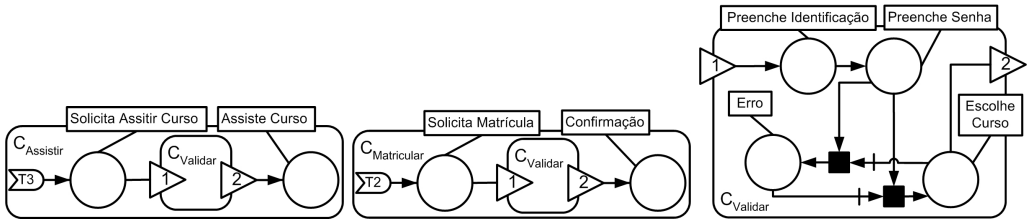
Após a realização da modelagem dos requisitos funcionais, modelos que capturem as características da estruturação interna do sistema, ou seja, sua arquitetura, devem ser construídos. Um aspecto importante na modelagem da arquitetura é que a mesma deve preservar o que foi definido como requisito para o sistema. De forma a ser possível a criação de mecanismos que assegurem tal preservação, uma Técnica de Especificação Formal (TDF) deve ser empregada para esse fim.

A escolha de uma TDF nessa modelagem se deve ao fato de ser possível criar regras formais para avaliar a concordância entre modelos especificados em níveis diferentes de abstração. Assim, considerando que os requisitos estão em um nível de abstração e que a arquitetura é um refinamento desse nível, é possível assegurar, por meio de simulação ou verificação formal [2], que a arquitetura obtida preserva o que foi definido como requisito funcional.

Dentre as possíveis TDFs que podem ser aplicadas em modelagens arquitetônicas (por exemplo SDL,



(a) diagrama de casos de uso.



(b) comportamentos dos casos de uso.

Figura 7: Diagrama fatorado de casos de uso e respectivos comportamentos.

LOTOS ou Estelle [21]), a AMBER foi escolhida por apresentar capacidade de simulação e verificação formal apoiadas por ferramentas [1] e por oferecer uma representação gráfica consistente, facilitando a definição dos modelos nas fases iniciais de projeto (requisitos).

A seguir é mostrado como os modelos de comportamento para os requisitos funcionais são transformados no modelo de comportamento da arquitetura inicial do sistema, e depois como esse modelo pode ser refinado. Para a construção do modelo de entidades basta mapear cada comportamento em uma entidade e colocar pontos de interação entre as entidades que representam comportamentos que executam atividades em comum (interações).

6.1 Transformação dos Requisitos na Arquitetura do Sistema

Da forma como foi definido na modelagem de requisitos, não há uma divisão de responsabilidades na execução das interações entre ambiente e sistema. Assim, o primeiro passo para a transformação desses modelos em modelos arquitetônicos consiste na divisão dessas responsabilidades entre entidades representando separadamente o ambiente e o sistema. Para a obtenção de tais entidades, dois passos devem ser seguidos: (1) primeiro os modelos comportamentais dos casos de uso devem ser compostos em um único modelo monolítico de comportamento; (2) depois tal modelo monolítico deve ser dividido em dois comportamentos, referentes ao ambiente e ao sistema.

Na composição, primeiro os comportamentos dos casos de uso que não apresentam inclusões devem ser agrupados sem um relacionamento explícito entre si. Após isso, regras devem ser seguidas para estabelecer o inter-relacionamento entre os comportamentos dos casos de uso que apresentam inclusões em comum.

Essas regras têm por finalidade evitar duplicações de unidades funcionais na definição da arquitetura do sistema, e por conseqüência evitar duplicações funcionais entre os possíveis componentes candidatos ao reuso já que a arquitetura será usada como base para a atividade de qualificação.

Essas regras são apresentadas na Tabela 1, onde o comportamento C é incluído (instanciado) nos comportamento $B_1, B_2, ..., B_n$. Maiores informações e detalhes sobre essas regras podem ser encontradas em [11].

A Figura 8 apresenta a aplicação das regras propostas na Tabela 1 sobre o comportamento $C_{Cadastrar}$ modelado na Figura 6 (a) e os comportamentos $C_{Assistir}$, $C_{Matricular}$ e $C_{Validar}$ modelados na Figura 7 (b). Observe que nesse diagrama os nomes das ações foram abreviados devido a problemas de legibilidade do gráfico. Vale salientar que para sistemas mais complexos esse diagrama de composição e os demais diagramas comportamentais para a arquitetura seriam inviáveis de serem construídos. Para contornar essa situação é

Tabela 1: Regras para a composição dos comportamentos dos casos de uso.

(1)	Um relacionamento de escolha deve ser posto entre os triggers dos comportamentos $B_1, B_2, ..., B_n$.
(2)	As relações de causalidade das ações do comportamento B habilitadas pelo ponto de entrada do mesmo são substituídas por uma disjunção formada pelas relações de causalidade que habilitam os pontos de entrada das instâncias de B em $B_1, B_2, ..., B_n$.
(3)	Para cada comportamento $B_1, B_2, ..., B_n$, a relação de causalidade das ações, que são habilitadas por um ponto de saída da instância de B, é trocada por uma conjunção entre a relação de causalidade que o ponto de saída do comportamento B e o trigger do comportamento em questão.
(4)	As demais relações de causalidade permanecem iguais.

possível utilizar a representação textual que AMBER oferece [17], deixando a representação gráfica apenas para a modelagem dos requisitos.

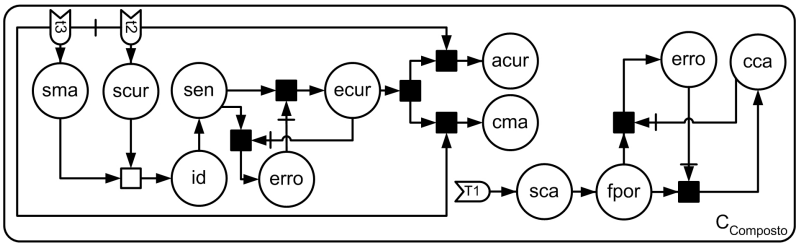


Figura 8: Composição dos comportamentos dos casos de uso.

Depois do comportamento monolítico ter sido obtido, o mesmo deve ser dividido de forma a representar separadamente o sistema e seu ambiente. Nessa divisão, as ações desse comportamento são transformadas em interações por meio da técnica de estruturação usando restrições[1] apresentada em [17].

Além da utilização da estruturação usando restrições, alguns passos devem ser seguidos para se obter uma correta divisão de responsabilidades entre ambiente e sistema. Esses passos são apresentados na Tabela 2. Maiores informações e detalhes sobre esses passos podem ser encontradas em [11]

Tabela 2: Passos para a divisão entre sistema e ambiente.

(1)	Os triggers devem ser postos no comportamento do ambiente, habilitando uma parte da interação. Caso algum trigger faça parte diretamente da relação de causalidade de alguma disjunção, o mesmo deve ser relacionado ao comportamento do sistema usando-se pontos de entrada e saída.
(2)	As disjunções obtidas pela aplicação da regra 2 da Tabela 1 devem ser postas no comportamento do sistema.
(3)	As conjunções obtidas pela aplicação da regra 3 da Tabela 1 devem ser transformadas em duas relações de causalidade, uma para a parte da interação que fica no ambiente e outra que fica no sistema, sendo a parte da interação que fica no ambiente habilitada pelo trigger correspondente, e a parte que fica no sistema habilitada pela outra relação de causalidade da conjunção.
(4)	Todas as demais relações entre ações devem ser postas no comportamento do sistema.

A Figura 9 apresenta a decomposição, por meio da aplicação da estruturação usando restrições conjuntamente com as regras estabelecidas na Tabela 2, do comportamento modelado na Figura 8. Nessa figura,

[1]A técnica de estruturação usando restrições baseia-se na decomposição de uma ação em uma interação, tal que a conjunção das condições de causalidade e restrições dessa interação seja igual à condição de causalidade e restrições da ação original.

os semicírculos ligados por linhas representam graficamente o conceito AMBER de interação.

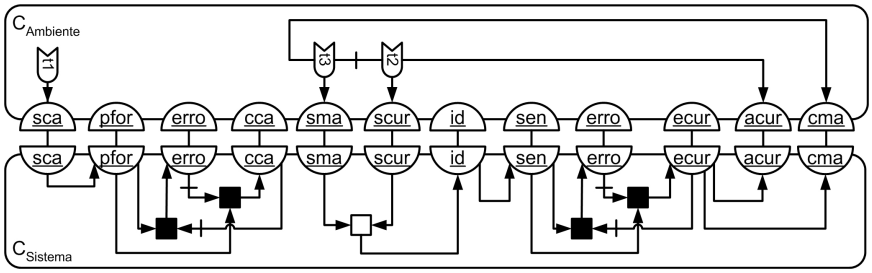


Figura 9: Divisão entre ambiente e sistema.

6.2 Refinamento da Arquitetura

Uma vez que a arquitetura inicial do sistema tenha sido obtida, representando o mesmo por meio de um modelo monolítico, essa arquitetura deve ser particionada de forma a representar as partes que o compõe. Nesse particionamento duas atividades são definidas: (1) primeiro o comportamento é refinado e avaliado; (2) para depois ser decomposto em partes.

No refinamento, ações e relações de causalidade são adicionadas ao comportamento de forma a representar estruturas internas relevantes a esse novo nível de abstração. Basicamente, dois tipos de refinamentos, ou uma combinação desses dois, podem ser aplicados: refinamento da causalidade e refinamento da ação [17].

Após o comportamento ter sido refinado, uma avaliação do mesmo deve ser realizada de forma a verificar se o comportamento obtido é equivalente ao comportamento original, isto é, se o mesmo preserva a funcionalidade prescrita. Técnicas e regras para a verificação dessa equivalência podem ser encontradas em [17], sendo que a princípio esse processo pode ser automatizado (computadorizado).

Caso o comportamento refinado seja equivalente ao comportamento original, o mesmo pode ser decomposto em múltiplos sub-comportamentos, representando as partes que comporão essa nova abstração da arquitetura. Para que a conformidade dessa decomposição seja garantida, as técnicas de estruturação usando causalidade[2] e estruturação usando restrições [17] devem ser aplicadas.

A Figura 10 (a) apresenta o refinamento do comportamento do sistema modelado na Figura 9. Nesse refinamento, as ações ckfo e val foram adicionadas e relacionadas às (inter)ações já existentes. A ação ckfo modela a checagem do formulário de cadastro do usuário e a ação val modela a validação da identificação e senha do aluno. A Figura 10 (b) apresenta a decomposição desse comportamento refinado em dois comportamentos $C_{Interface}$ e $C_{ControledeDados}$.

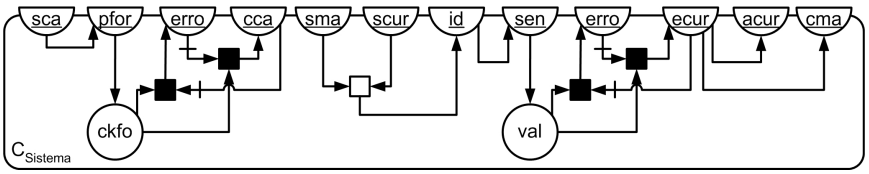
7 QUALIFICAÇÃO DE COMPONENTES

Uma vez que a arquitetura do sistema tenha sido refinada, uma tentativa de povoamento da mesma com componentes pré-existentes deve ser feita. A essa tentativa é dado o nome de qualificação de componentes. Na qualificação, a escolha dos componentes que comporão o sistema é feita por meio de uma comparação entre as partes definidas na sua arquitetura e as descrições dos componentes pré-existentes.

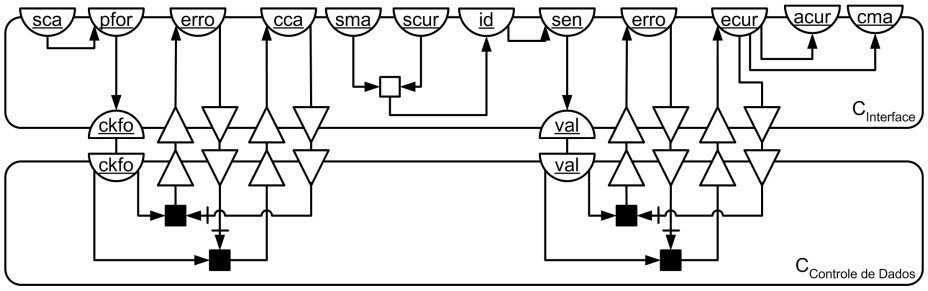
Uma restrição da abordagem proposta é que os componentes pré-existentes devem ser classificados (descritos) usando-se modelos de comportamento AMBER. Assim, mecanismos formais podem ser criados de forma a assegurar que os componentes candidatos ao reuso irão realizar as funções (ou parte delas) definidas na arquitetura.

Se após a qualificação ainda existirem partes da arquitetura que não foram assinaladas a componentes, as mesmas podem ser refinadas e decompostas (procedimento apresentado na seção 6.2), e a qualificação pode ser novamente aplicada sobre as partes obtidas.

[2]A técnica de estruturação usando causalidade baseia-se na decomposição de relações de causalidade, permitindo que uma ação e sua condição de ocorrência sejam definidas em sub-comportamentos distintos.



(a) Refinamento.



(b) Decomposição.

Figura 10: Refinamento e decomposição do comportamento.

7.1 Escolha dos Componentes Pré-existentes

De forma a verificar quais partes da arquitetura podem ser implementadas por meio de componentes pré-existentes, todos os modelos de comportamento referentes a essas partes são comparados aos modelos de comportamento que descrevem tais componentes.

Essa comparação se baseia no conceito de equivalência observacional proposta em [10]. Segundo esse conceito, dois comportamentos são considerados equivalentes sempre que não for possível diferenciá-los a partir de uma perspectiva externa. Aqui, a avaliação dessa equivalência pode ser dividida em duas etapas: (1) geração das seqüências externas de execução dos comportamentos; e (2) comparação dessas execuções entre si de forma a verificar a equivalência entre os mesmos. Um modelo de execução para comportamentos AMBER pode ser encontrado em [17].

Para se obter as seqüências externas de execução para um determinado comportamento, primeiro todas (ou uma grande parte) as possíveis seqüências de (inter)ações que podem ocorrer no mesmo devem ser geradas. Depois suas ações internas devem ser abstraídas dessas seqüências, restando-se apenas as interações externamente visíveis.

Uma vez que essas seqüências tenham sido obtidas, a verificação de equivalência deve ser realizada. Nessa verificação, o comportamento C_a é considerado equivalente ao comportamento C_b, quando as seqüências de interações estabelecidas em C_a englobarem as seqüências estabelecidas em C_b, obedecendo a mesma ordem de ocorrência.

Supondo que em um dos passos de refinamento da arquitetura do sistema uma das partes obtidas tenha as seqüências de execução apresentadas na Figura 11 (a), e um dos componentes pré-existentes tenha as seqüências de execução apresentadas na Figura 11 (b), aplicando-se a atividade de qualificação sugerida é possível afirmar que tal componente pode ser selecionado para implementar tal parte, já que a seqüência de execução 1 dessa parte é englobada pelas seqüências de execução 1, 2 e 3 do componente e a seqüência de execução 2 dessa parte é englobada pelas seqüências de execução 2 e 3 do componente, na mesma ordem de ocorrência.

Na verdade o processo de qualificação sugerido é um processo de pré-seleção de componentes que visa agilizar e semi-automatizar a qualificação de componentes, não configurando o mesmo um processo completo em si. Para que a qualificação seja completa, outras informações devem ser levadas em consideração na seleção/classificação dos componentes. Porém, como essas informações não foram consideradas no processo sugerido, as mesmas não serão aprofundadas, apenas fica indicado o *modelo 3C* (conceito, conteúdo e contexto) definido em [20] para tal finalidade.

1. Identificação - Senha
2. Identificação - Senha - Erro

(a) seqüências de execução para uma parte da arquitetura.

1. Identificação - Senha - Escolher Curso - Exibir Curso
2. Identificação - Senha - Escolher Curso - Erro
3. Identificação - Senha - Erro
4. Identificação - Erro

(b) seqüências de execução para um componente pré-existente.

Figura 11: Seqüências de execução.

8 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou uma abordagem que visa enriquecer o processo DBC por meio da inserção de um sub-processo baseado em refinamentos sucessivos. Nesse sub-processo o modelo arquitetônico formal AMBER foi usado conjuntamente com diagramas de casos de uso na modelagem dos requisitos funcionais do sistema, subseqüente transformação desses modelos em modelos para sua arquitetura e final pré-seleção dos componentes candidatos para compor o mesmo.

Na modelagem dos requisitos do sistema, casos de usos foram utilizados em combinação com modelos de comportamento AMBER. já na modelagem da arquitetura, modelos de comportamento e entidade AMBER foram empregados. Como a utilização de casos de uso permite uma melhor estruturação do comportamento do sistema e tais modelos servem como base para a definição de sua arquitetura inicial, tal arquitetura pode ser melhor estruturada, reduzindo duplicações funcionais que possam ocorrer.

Além disso, uma vez que um modelo formal é aplicado nessa abordagem, é possível criar mecanismos que assegurem que dois modelos especificados em níveis diferentes de abstração estão em conformidade. Assim, considerando que a arquitetura é um refinamento dos requisitos e tendo a atividade de qualificação baseada nessa arquitetura, é possível assegurar que os componentes selecionados preservam o que foi prescrito como requisito funcional para o sistema (ou pelo menos uma parte deles), sendo possível semi-automatizar tal atividade.

Como trabalho futuro, poderia ser criado um mecanismo de ajuste do modelo arquitetônico que o adaptasse de forma a refletir no mesmo o reuso dos componentes selecionados. Nesse ajuste, os modelos que representam os componentes substituiriam os modelos para a(s) parte(s) da arquitetura que os mesmos implementam. Assim, essa nova arquitetura poderia ser simulada de forma a verificar se por meio da reutilização desses componentes sua funcionalidade é preservada.

Por fim, outro ponto importante que pode ser estudado diz respeito a mudanças de requisitos e como essas mudanças refletiriam no modelo da arquitetura e por conseqüência na atividade de qualificação.

Referências

[1] BiZZdesign. Bizzdesigner tool. Endereço: http://www.bizzdesign.nl, 2005.

[2] T. Bolognesi, J. Lagemaat, and C. A. Vissers, editors. *LOTOSphere: Software Development with LOTOS*. Kluwer Academic Publisher, 1995.

[3] S. R. Christensen. Software reuse initiatives at lockheed. *CrossTalk*, 8(5):26–31, 1995.

[4] D. F. D'Souza and A. C. Wills. *Objects, Components and Frameworks with UML: the Catalysis Approach*. Addison Wesley, USA, 1999.

[5] C. R. G. Farias, L. F. Pires, and M. van Sinderen. A component-based groupware development methodology. In *In Proceedings of the 4 Int. Enterprise Distributed Object Computing Conferece*, pages 204–213, 2000.

[6] C. R. G. Farias, M. van Sinderen, and L. F. Pires. A systematic approach for component-based software development. In *Proceedings of Seventh European Concurrent Engineering Conference (ECEC'2000)*, pages 127–131, Leicester (United Kingdom), April 2000.

[7] J. J. Jeng and B. H. C. Cheng. Using formal methods to construct a software library. In *In Proceedings of 4th European Software Engineering Conference, Lecture Notes in Computer Science*, volume 717, pages 397–417, September 1993.

[8] A. Mili, R. Mili, and R. Mittermeir. Storing and retrieving software components: A refinement based system. In *In Proceedings of 16th International Conference on Software Engineering*, pages 91–100, Sorreno, Italy, May 1994.

[9] Object Management Group (OMG). Unified modeling language specification: Version 2.0. Endereço: www.uml.org, 2005.

[10] D. Park. Concurrency and automata on infinite sequences. In LNCS 104, editor, *5th GI Conference*, pages 167–183, 1981.

[11] F. V. Paulovich. Uma abordagem para o desenvolvimento de sistemas distribuídos baseada em componentes. Master's thesis, Universidade Federal de São Carlos, São Carlos/SP -Brasil, Fevereiro 2003.

[12] J. Penix. *Automated Component Retrieval and Adaptation Using Formal Specifications*. PhD thesis, April 1998.

[13] L. F. Pires and C. R. G. Farias. Amber: uma linguagem para o desenvolvimento de sistemas distribuídos. In *XIX Simpósio Brasileiro de Redes de Computadores*, pages 82–97, Florianópolis, Brasil, 2001.

[14] G. Pour. Component-based software development approach: New opportunities and challenges. *Technology of Object-Oriented Languages*, pages 375–383, 1988.

[15] G. Pour, M. Griss, and J. Favaro. Marking the transition to component-based enterprise software development: Overcoming the obstacles - patterns for success. *Technology of Object-Oriented Languages and Systems*, 1999.

[16] R. S. Pressman. *Engenharia de Software*. McGraw Hill, 5 edition, 2002.

[17] D. A. C. Quartel. *Action relations: basic design concepts for behaviour modelling and refinement*. PhD thesis, University of Twente, Enschede, the Netherlands, 1998.

[18] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

[19] C. Szyperski. *Component Software - Beyond Object-Oriented Programming*. Addison-Wesley, 1998.

[20] W. Tracz. Where does reuse start? In *Realities of Reuse Workshop*. Syracuse University CASE Center, 1990.

[21] K. J. Turner. *Using formal description techniques - An introduction to Estelle, Lotos and SDL*. John Wiley and sons, 1993.

[22] C. A. Vissers, M. J. van Sinderen, and L. F. Pires. What makes industries to believe in formal methods. In *Protocol Specification, Testing and Verification XIII*, pages 3–26, Netherlands, 1993. Elsevier Sciende Publishers B. V. (North-Holland).

Ugatze : Model Driven Engineering for Component Reuse

P. Aniorté, F. Seyler
LIUPPA
IUT de Bayonne – Département Informatique
Place Paul Bert
64100 Bayonne – France
{aniorte,seyler}@iutbayonne.univ-pau.fr

Abstract

This paper relates to the engineering of heterogeneous distributed systems based on the reuse of software components. These are high level software components considered as autonomous in terms of run-time. Our research field deals with several research domains: distributed systems, reuse, interoperability and components. In this paper, we propose a metamodel for reuse of components, called Ugatze. First, we present those different research fields, and we position our component metamodel with regard to the state-of-the-art. Next, we present the two viewpoints which constitute of this component metamodel, which are respectively a component interface and a set of conceptual integration tools. Our proposal in this article is based on meta modeling standards. The last part illustrates with an example a real software application built with Ugatze components and an application of our reuse process.

Keywords: Information system, Reuse, Component metamodel, Integration, Interoperability

Resumen

Los trabajos presentados se refieren a la ingenieria de sistemas distribuidos y heterogeneos basada en la reutilizacion de componentes. Se trata de componentes de software de alto nivel en el sentido que estan autonomos en termino de run-time. Nuestro campo de investigacion se ubica a la interseccion de varios ambitos : sistemas distribuidos, reutilizacion, interoperabilidad, componentes. En este articulo, el nucleo de nuestra propuesta es un metamodelo, llamado Ugatze, que permite la especificacion y la integracion de componentes que reutilizar ("legacy" componentes). Primero, presentamos los varios ambitos vinculados con nuestros trabajos. El objetivo consiste en ubicar nuestra propuesta. Luego, describimos los dos puntos de vista de nuestro modelo de componentes, es decir la especificacion de los componentes con interfaces y su integracion con un conjunto de herramientas conceptuales. Nuestra propuesta esta basada on estandardes de metamodelizacion y ilustrada con un caso desarrollado dentro de un proyecto de la Union Europea.

Palabras claves: Sistemas de informacion, Reutilizacion, Metamodelo de componentes, Integracion, Interoperabilidad

1. INTRODUCTION

We are interested in and devoted to Information Systems (IS) engineering based on reuse. The objective is to treat the multiplication and the dissemination of heterogeneous information [20]. Our approach is consistent with some research orientation developped in « High Confidence Software and Systems » [3] which is based on the « component » paradigm for the reuse of existing software components and their integration in a distributed application. Through its principles of decoupling between entities and encapsulation, the component paradigm seems appropriate to reuse [2]. In the component model Maleva [5], components cooperate by information stream and appropriately by control interactions. This concept which is called separation between data stream and control stream, and has been introducted in SADT [7] in another context. Consequently, serveral reasons which are described in section 2, lead us to model our proposal according to tools and standard specifications proposed by the OMG : UML [12] MOF [13], OCL [22] and MDA [21]. Our main focus is based on reuse from which our component meta model is articulated around two interlaced viewpoints that are supposed to provide only one at the end of the integration process. These two viewpoints support two different concerns :

•The component aspect conceptualizes the activity of components specification and adaptation. It is symbolised by the MOF package « Ugatze : :Component », described in section 3.

•The integration aspect models the integration infrastructure by the intermediary of generic tools provided in order to allow such an integration. This viewpoint is symbolized by a MOF package « Ugatze::Interactions » described in section 5.

The integration of these two viewpoints will provide a component/interaction model which provides an architecture independent from all execution platform. According to MDA specifications [21], this architecture is a PIM (Platform Independant Model). The construction and the manipulation of the application model is treated in section 5.

2. STATE OF THE ART

Our works integrate several research areas : distributed Information Systems, reuse, interoperability, component paradigm. This part etablishes short state of art in order to position our works.

2.1 Systems Engineering and reuse

Reuse [4][10] is defined as a system integration approach allowing to construct systems from existing elements. The goal of reuse component engineering is to produce an infrastructure of reuse which exploits a component's representation through the abstraction principle which separates the component specification and their real reusable part. Distributed applications are currently targeted towards multiple communication modes. There are « server » side applications developed by objects oriented middleware (CORBA or EJB), or message oriented middleware, and even some middleware scrutinizing the data continuous stream. The proliferation of middleware gradually takes over the proliferation of languages, so that the integration work passes significantly through the establishment of footbridges between middlewares.

2.2 The Component Paradigm

The term « component » is widely used in the reuse area, with a general idea of reusable entity. Primarily, such approaches are based on de-coupling between components. The aim and objective is to express external dependancies between components [17]. Secondly, these approaches propose some communication mechanisms. For example, in the MALEVA [9] component model, components use some communication terminals to communicate with other components. Each component can be specified by its interface. Among commercial components model, one can note EJB, a partial component model which mainly proposes « offered interfaces », or Corba Component Model, which proposes some interaction ports: event sources and wells, offered and required interfaces. Regarding the references in relation to abstract components model, the ODP computational viewpoint [18] proposes interfaces according to three communication models such as synchronous, asynchronous and continuous stream.

2.3 Meta Modeling

2.3.1 Definition of meta models

Underlying the specifications of OMG, two views seem analogous to allow the building of meta models. The « UML profile » approach [12] which consists in extending the UML meta model by adding some stereotypes, constraints and tagged values, adapting the UML meta model to a particular application area. The MOF approach consists in building some MOF meta models by using (instanciate) concepts of the MOF Meta Meta Model, commonly called MOF Model [16]. The MOF specification describes a set of modeling elements used to define the structure, the semantic and the constraints of meta models. Similarly to UML specification, constraints are expressed using OCL (Object Constraint Language) [22].

If these two approaches are equivalent, the prior advantage is to use UML representations (for instance : class diagrams) only to represent the MOF model's concepts, without associating the entire UML semantic. Many UML profiles are currently available, particulary EDOC [15] and EAI [14].

2.3.2 Exploitation of models : Model Driven Architecture

However, the representation of a meta model is a sole part of a broad work environment which treats models as first class entities: Model Driven Architecture (MDA). MDA was proposed by the OMG as a response to the « middleware proliferation », middlewares which can be standard (CORBA from the OMG), industrial (COM+, .NET) or « somewhere in the middle » (HTTP/XML/SOAP). Indeed, it is unfeasible for a wide company to envisage the use of a sole middleware, for economical and technological reasons. Widely, spread products such as CORBA, EJB, XML/SOAP, COM+ and .NET do exist. MDA proposes to build a « stable » model, independent from the middleware and based on the OMG modelisation standards : UML, CWM (Common Warehouse Meta-model) and MOF. Its objective is to capitalize and reuse this model instead of continuously migrating from one middleware to another. The major MDA innovation is to specify a set of model transformations : from the more abstract model (Platform Independant Model) to the concrete ones, taking into account the middlewares (Platform Specific Model or PSM), by going through intermediary steps : transformation of PIM towards PIM, PIM towards PSM, PSM towards PIM… The major benefit for a reuse process based on components is that it is supported by the existing standards in term of representation, verification and manipulation of models (UML, OCL, MOF).

3. COMPONENT INTERFACE VIEWPOINT

Underlying the introduction (refer to section 1), the component model UGATZE is split into two vierwpoints. The interface viewpoint is a set of abstraction criteria and used constraints allowing to specify components independently from any platform. This viewpoint is derived from MOF package which defines a naming space for the introduced concept and illustrates the following principles: abstraction, decoupling between components, variability, separation between data and control, and multiplicity of communication modes (refer to Figure 1).

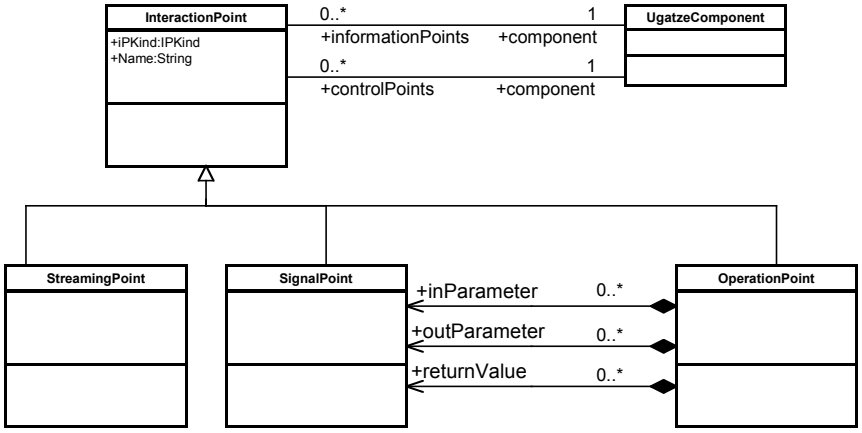


Figure 1 Component Interface

The component is split into two parts: specification and realization. The specification part is expressed by components' interface and modelled by the « UgatzeComponent » MOF class. The abstraction results from an activity of retro engineering of existing applications, which is an ascending approach. The decoupling between these components shows that the dependencies among the components are defined externally to these ones. The dependencies rely on the interaction points (« InteractionPoint » MOF class). The principle of variability takes into account that the component has a fixed part and a variable one which guarantees the component genericity. Ugatze gives also three major interaction modes such as the asynchronous mode (signal points), the synchronous mode (operation points) and the continuous stream (streaming points).

Asynchronous mode indicates what the component « receive » or « transmit » to its environment. Asynchronous data supports the events (whose sole occurrence has semantic), or the messages (atomic elements with complete content).

Asynchronous mode and streaming mode indicate a one-way communication, so that we designate interaction points by « transmitter » (Out) or « receiver » (In). The synchronous mode holds a central place within the « object » world and the « technological components » world. This induces both a passage of data and control. In Ugatze, the operation point is seen as a « composite » interaction point, which is specified by its input and output points. Each operation point, designates an « offered » service and a « required » as in standard component models or factual standards (CCM and lesser extent EJB). As a result, we are no longer in a unique input/output characterization. Components have several interaction points which designate services.
The Ugatze meta model primarily proposes two sets of interaction points: information points and control points with similar global operation (in term of diffusion type and procedure) but different ending (refer to Figure 2). The membership of this set is characterized by the « kind » attribute of the « InteractionPoint » class.
We also distinguish two diffusion types which are unicast (point to point) and multicast (refer to section 4). In a general way, multicast requires the intervention of a third device (e.g. publication/subscription or group) whilst unicast does not require any.

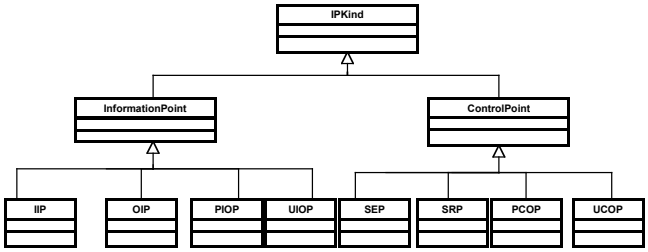


Figure 2 Kinds of interaction points

Among information points, Input Information Points (IIP) are interaction points on which components recover the one-way data flow. We will speak about IIP for a streaming point or a signal point. We also find some Output Information Points (OIP), which play the symetrical role with uploading the data one-way flow towards the component outside. The Provided Information Operation Point (PIOP) responds to requests, and finally, the Used Information Operation Point (UIOP) determines services required for the execution of a Ugatze component.
In MALEVA, the separation between data and control flows has already been proposed. However, this separation concerns only the activation or desactivation of component: a component shifts control once its « work » is finished. This induces a strong assumption on the component state (activated/desactivated). In Ugatze, no assumption is based on this state and the way of flow creation, use or transformation. Ugatze proposes a set of control point which either allows a component to synchronize another component or to be synchronized. These interaction points are called SEP (Signal Emission Point) and SRP (Signal Reception Point). The separation between functional part and life cyle of the components is also a principle that leads the server side components such as CCM and EJB. These models use various devices as the « Home » abstraction or the « Factory » abstraction to symbolize the life cycle in the code. Ugatze also includes this possibility through different operation points: Provided Control Operation Point (PCOP) for offered control operations and Used Control Operation Point (UCOP) for required control operations.
Ugatze meta model is completed by a set of constraints expressed in OCL and stored in a package.

4. INTERACTION VIEWPOINT: COMPONENTS INTEGRATION

The components to be reused are intended for integration in order to develop a new system. Within this system, the components must be able to interoperate. Interoperability is described at a conceptual level. In an abstract way, the viewpoint « Interaction » defines the integration tools. This part of the Ugatze meta model, represented by a MOF package, is separated from the Interface viewpoint because it aims at a different concern: the integration of components.
ASIMIL (Aero user-friendly SIMulation-based dIstant Learning) is a european project which aims to improve the aeronautical team's training [1] [11]. The system, based on intelligent agents, is made up of different entities. Several languages (Java, C, C++) and different environments (Linux, Windows) consitute the support of theses entities. Each of these entities is seen as a component that must be reused to develop the system in a distributed environment. Our contribution to the ASIMIL project consisted of integrating these different components. In consequence and evidence, it proves to be an excellent application field for our research works. The integration of ASIMIL components based on simple examples allows to illustrate our propositions regarding interactions.

Interactions are « basic » abstract tools which belong to the meta model and allows a conceptual integration of autonomous components. These tools are addressed to « real time » oriented components with control interaction. EAI area is also addressed by asynchronous interaction allowing communication by « message » and a decoupling in term of execution between components. The integration by Ugatze tool is not limited to communication but also concerns the control. In its most abstract form, an interaction connects a certain number of interaction points (at least 2). As related to the interaction point, interactions are categorized into subclasses, by simple heritage. In order to avoid classes proliferation, we have introduced the « kind » attribute which allows to qualify interactions according to interaction points, with the help of OCL constraints (refer to Figure 3).

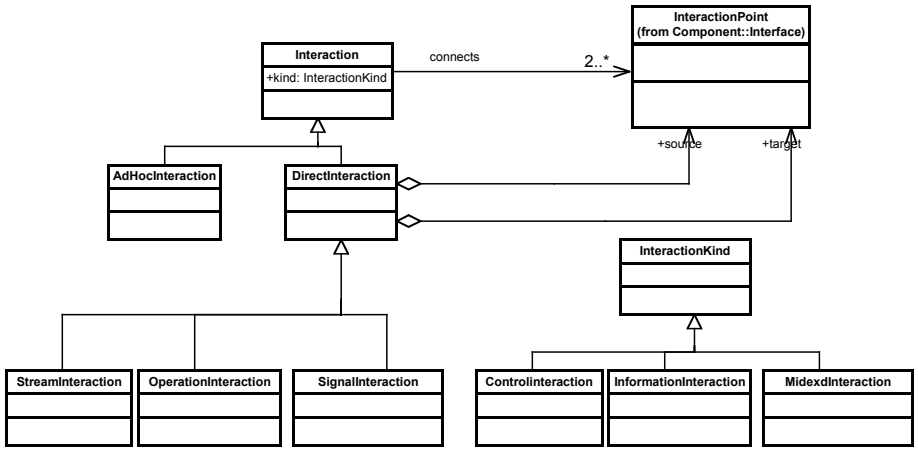


Figure 3 : Interactions

4.1 Direct interaction : definition, constraints and graphic representation

Direct interaction designates a direct connection between two (and strictly two) interaction points). The mode of communication mode is given by the connected interactions points. Thus, a data flow interaction designates the connection between two data points (refer to Figure 4). An operation interaction designates an interaction between two operation points which offer compatible signatures (input, output and return parameters). Finally, constraints concern the multiplicity and the characterization of interaction points : a « source » and a « target », the conveyed data type…

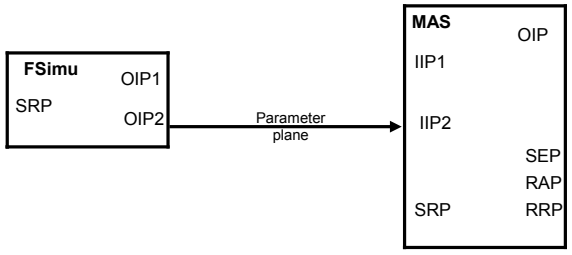


Figure 4 : Direct data interaction (Illustration from the ASIMIL european project)

The interaction concept is similar to the « binding object » concept as specified by the « treatment » aspect of RM-ODP [18], or by the « connector » aspect in the ADL's area [8]. The direct data interaction only concerns an IOP and an IIP, which conveys the same data type. For example, direct data interaction allows to connect a component's IOP to another one's (IIP) [19]. In this manner, the component « Multi Agent system » (MAS) exploits the continuous stream of flight parameters from the flight simulator component (FSimu).

Direct control interaction connects an SEP and a SRP, with some typing constraints on the conveyed control information. In the example below (refer to Figure 5), the PFC (Procedure Follow up Component), an electronic checklisting , remotely launches the flight simulator's execution (FSimu).
In the same way, data operation interaction connects a PIOP (Provided Information Operation Point) and a UIOP (Used Information Operation Point), with some typing constraints on connected interaction points. Control operation interaction connects a PCOP (Provided Control Operation Point) and a UCOP (Used Control Operation Point), with also some typing constraints on the connected interactionpoints.

Figure 5: Direct Control Interaction

4. 2 Ad hoc integration mechanisms: ad hoc interactions

Ad hoc interactions allows a functional interoperability for heterogenous component to cooperate at a conceptual level, with a total abstraction from the technical platforms and the programming language. Ad hoc interactions are categorized hierarchically into three groups determined by the « kind » attribute (refer to Figure 3): complex information interaction (refer to Figure 6), complex control interaction and mixed interaction.

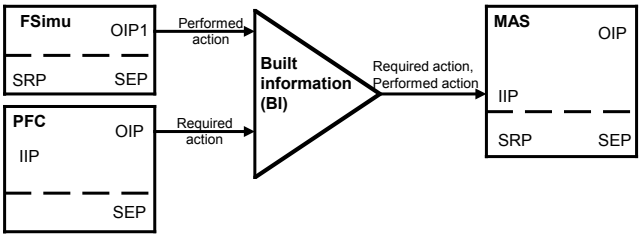


Figure 6: Complex information interaction between 3 components (Illustration from the ASIMIL european project)

• In accordance to an algorithm, complex information interaction consists in producing either one or several information based on a single or several information from a unary or multiple component. Of course, such interactions are also governed by OCL constraints that characterize connected interactions points. In Figure 6, the MAS component performs a Built Information (BI): required action, performed action. This information is made up from two components; Fsimu provides the action performed by the trainee while the PFC provides the required action according to the checklist.

• Complex control interaction is managed in the same way. The sole difference lies in the connected interaction point, which are control interaction points. In other terms, it allows to generate a new control element from control elements. This strategy is particulary used in the real time world.

• Mixed interaction is the generalization of the two previous items. More precisely, it aims to produce an information element or a control element from information or control elements.

4. 3 Pre-developed integration mechanisms : preset components

Most of the component platforms propose preset tools (or services) that take the form of preset and easily accessible components. The best example is CORBA and its associated services. Ugatze proposes a certain number of preset tools pre-defined in the form of component which facilitates applications development by components reuse.

Multicast mechanism

Among the properties of interaction points proposed by Ugatze, it is advisable to notice that interaction points can be qualified as « unicast » ones, i.e. that they do not « care » about the number of receiver components. Multicast mechanism allows a component to send the same data to several receivers connected to « multicast » by mean of a

special interaction. Conceptually, this mechanism takes the form of an ad hoc interaction which has an IIP and several OIP, related to the same data type.

Mailbox
The mailbox mechanism is used when an information is not transfered directly by a component to another component, but into the box while waiting for another component to claim for it. Regarding the deposit in this mailbox, a direct interaction is used to connect the transmitter component (through an OIP) with the mailbox. The removal is materialized by an interaction between the mailbox and an IIP of the connected component.
This mechanism is illustrated in Figure 7. The MAS component broadcasts an information to the PFC and the mailbox, while a third component (the History Manager – HM component) has the responsability to periodically recover it.

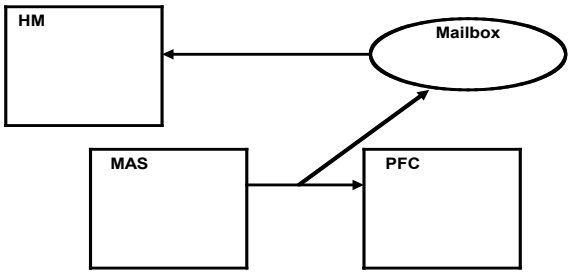


Figure 7: Mailbox and multicast mechanism (Illustration from the ASIMIL european project)

Sharable resource
Sharable resource is a typical example of a control mechanism. In the assumption where several components use the same resource, each implied component must gain access to the resource in turn and release it when its execution is completed. Each component has a Resource Access Point (RAP) and a Resource Release Point (RRP). This mechanism offers a solution to problems generated by simultaneous existence of several competive components. In fact, RAP and RRP are two special Signal Emission Points (SEP) allowing components to allocate and release sharable resources.

5 EXPERIENCE FEEDBACK

According to MDA specifications, the « Ugatze » meta model instance, built using the Interface viewpoint (refer to section 3) and the Interaction viewpoint (refer to section 4), is a PIM (Platform Independant Model). A graphical representation is associated to the PIM which represents the software architecture of the heterogeneous distributed system to develop (refer to Figure 8). The ASIMIL application, introduced in the previous section results from a conceptual integration process.

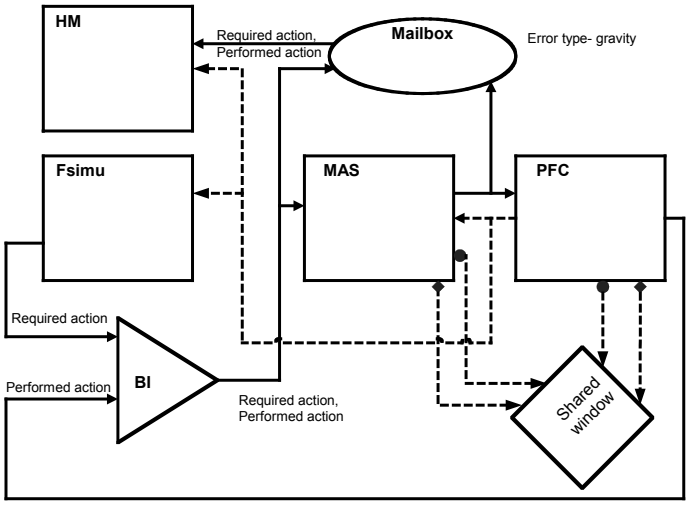


Figure 8: PIM of the ASIMIL european project

This graph models the integration of the components described in the previous section. One can also note that the whole kinds of interaction shown in section 4 are used in this graph: direct interactions, ad hoc interactions and preset components (multicast, mailbox and sharable resource). The component interface interaction points are not necessarily used. For instance, it is the case of the SEP of the MAS component (refer to section 4). Each component is reusable and adaptable (principle of variability) to different applications, according to the use of interaction points. The graph construction is basically the first step of the reuse process. This step builds an architectural view, independent from platforms and programming language.

The exploitation of another viewpoint (Technical Viewpoint), not described in this paper, allows to introduce elements linked to the implementation of the conceptual software architecture: operating system, language, middleware… (refer to Figure 9).

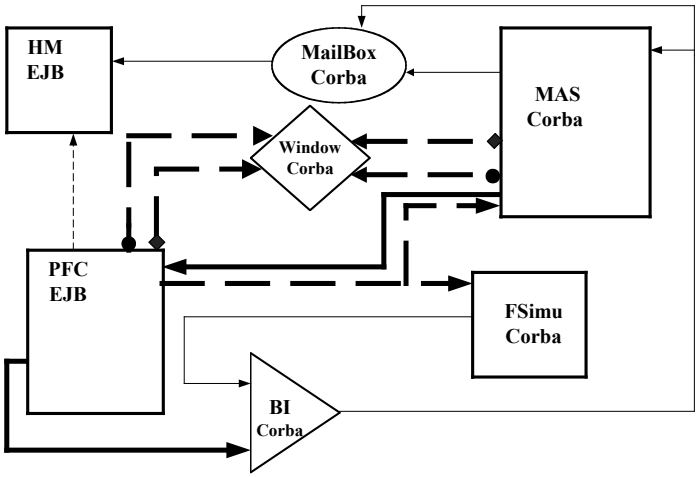


Figure 9: PSM and M2M interactions of the ASIMIL european project

In a graphical illustration, this step exploits a graph's transformation emphasizing « inter middleware » interaction. Considering our application, such interactions also called M2M (Middleware to Middleware) interactions link Corba components and EJB components. Each set of components using the same platform is a PSM (Platform Specific Model) in MDA terminology. (see figure 9). Regarding MDA, the originality of ours works lies in the fact that we

simultaneously derived several PSM from one PIM. The on going research focuses on the modelization and the implementation of M2M interactions using footbridges.
Technicaly speaking, the Ugatze integration process is based on the MDA specification, and in particular on a MOF referential allowing to reificate conformed models to the Ugatze meta model. The reification of these models allows their manipulation. Actually, the implementation of automatic transformation tools is the object of an investigation, while the entire integration process has been validated with « human » transformations. The ASIMIL european project also give us the opportunity to validate the implementation of all the three kinds of interactions (refer to the subsections of section 4) as well as various footbridges between different platforms. For instance, an interoperability bridge has been « thrown » between the simulator (Fsimu component), implemented in C++ on a Corba 2.3 platform, and the Multi Agents System (MAS component) implemented with EJB (Message Driven Bean). Regarding the ad hoc interactions, a Java/Corba component allows the implementation of the « BI » interaction (refer to Figure 9), using the Corba Notification Service (CNS) to produce the required information.

6. CONCLUSION

In this paper, we have presented a component meta model, called Ugatze, allowing the specification of the distributed heterogeneous components and their conceptual integration. This meta model is based on two points of viewpoints. The « Interface » viewpoint allows to specify the interface of each component by the way of different interaction points supporting various important principles. The « Interaction » viewpoint allows the conceptual integration of the components to reuse using a set of interactions suited to the adressed issues. A MDA-based process with different transformation of models (PIM, PSM…) is associated with the Ugatze meta model allowing the effective implementation of the conceptual software architecture. The Ugatze meta model supports the separation of concerns corresponding to the different viewpoints relies on well-known standards of the OMG (MOF, OCL…).

REFERENCES

[1] P. Aniorte and F. Seyler. A component model to build a distributed software architecture. In *The 2nd International Conference on Information Systems and Engineering ISE 2002*, volume 34 of *Simulation Series*, pages 149–154, San Diego, July 2002. www.iutbayonne.univ-pau.fr/seyler/ISE2002.ps.

[2] C. Cauvet, D. Rieu, A. Front-Conte, and Ph. Ramadour. Réutilisation en ingénierie des systèmes d'information. In *Ingénierie des systèmes d'information*, pages 115–148. Hermès, 2001.

[3] HCSS. Hcss research needs : a white paper. Technical report, IWG/IT R& D, White House National Science and Technology Concil, 2001.

[4] K. Kang, S. Cohen, J.Hess, W. Novak, and S. Peterson. Featured-oriented domain analysis (Foda) feasability study. Technical report, Sotware Engineering Institute, Canergie Mellon University, Pittsburgh, Pennsylvania, 1990.

[5] Lhuiller. *Une approche à base de composants logiciels pour la conception d'agents*. PhD Thesis, Université Paris 6, February 1998.

[7] David A. Marca and Clement L. McGowan. *SADT: structured analysis and design technique*. McGraw-Hill, Inc., 1987.

[8] Nemad Medvidovic and Richard N. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26(1):70–93, January 2000.

[9] T. Meurisse and J.P. Briot. Une approche à base de composants pour la conception d'agent. *Technique et Science Informatique*, 20(4/2001):567–586, 2001.

[10] Hafedh Mili, Fatma Mili, and Ali Mili. Reusing software: Issues and research directions. *IEEE Trans. Software Eng.*, 21(6):528–562, 1995.

[11] A. Minko. *Evaluation Qualitative des Erreurs dans un Système Tuteur Intelligent basé sur la Simulation*. PhD Thesis, Université de Pau et des Pays de L'Adour, December 2002.

[12] OMG. Unified modeling language specification, version 1.4. Technical report, Object Managment Group, 2001. Object Managment Group.

[13] OMG. Meta object facility specifications. Technical report, Object Managment Group, March 2002. version 1.4.

[14] OMG. Uml profile and interchange models for enterprise application integration (EAI) specification. Technical report, Object Managment Group, 2002. final adopted specification.

[15] OMG. Uml profile for enterprise distributed object computing (EDOC). Technical report, Object Managment Group, 2002.

[16] M. Peltier. *Techniques de transformation de modèles basées sur la méta-modélisation*. PhD thesis, University of Nantes, 2003.

[17] F. Peschansky and al. Les composants logiciels : Evolution technologique ou nouveau paradigme ? In *Conférence OCM*, pages 53–65, 2000.

[18] J.R. Putman. *Architecting with RM-ODP*. Prentice Hall, 2001.

[19] M. Shaw and D. Garlan. *Software Architecture: Perspective on an Emerging Discipline*. Prentice Hall, 1996.

[20] N. Singh. Unifying heterogeneous information models. *Communications of the ACM*, 41(5):37–44, 1998.

[21] Richard Soley. Model driven architecture. Technical report, Object Managment Group(OMG), November 2000. White Paper Draft 3.2 (www.omg.org/mda).

[22] Joe Warmer and Anneke Kleppe. *The Object Constraint Language : Precise Modeling with UML*. Addison-Wesley, 1998.

Gerenciamento de Riscos para Projetos de Software

Pascale C. Rocha
pascale.mia@unifor.br

Arnaldo D. Belchior
belchior@unifor.br

Universidade de Fortaleza - UNIFOR
Mestrado em Informática Aplicada – MIA
Av Washington Soares, 1321
CEP 60.811.341 – Fortaleza – CE – Brasil

Abstract

Software development is a complex activity, involving several factors, which are sometimes unpredictable and hard to control. This complexity causes schedule delays, cost overruns, quality problems and missing functionality. As uncertainty is inherent to this kind of projects, risk management has become more notable as it leads the organization to systematically plan, anticipate, and mitigate risks to proactively eliminate or minimize their impact on the project. This paper suggests a software risk management process based on concepts of the RUP (2003), on risk approaches of CMMI-SW (2002), PMBOK (2004), IEEE STD 1540-2001 and AS/NZS 4360 (2004).

Keywords: RUP, PMBOK, CMMI, IEEE STD 1540-2001, AS/NZS 4360, Risk Management.

Resumo

O desenvolvimento de software é uma atividade complexa, envolvendo inúmeros fatores, muitas vezes imprevisíveis e difíceis de serem controlados. Esta complexidade faz com que grande parte dos projetos de software exceda o prazo e o orçamento previstos, além de não atender às expectativas de seus clientes em termos de funcionalidade e qualidade. Como a incerteza é inerente a este tipo de projeto, o gerenciamento de riscos vem-se tornando cada vez mais relevante, pois conduz a organização à sistematicamente planejar, antecipar e mitigar os riscos para proativamente eliminar ou minimizar seus impactos no projeto. Este trabalho propõe um processo para o gerenciamento de riscos de projetos de software fundamentando-se em conceitos do RUP (2003), nas abordagens de risco do CMMI-SW (2002), do PMBOK (2004), do padrão IEEE STD 1540-2001 e da AS/NZS 4360 (2004).

Palavras-chave: RUP, PMBOK, CMMI, IEEE STD 1540-2001, AS/NZS 4360, gerenciamento de riscos.

1. INTRODUÇÃO

O desenvolvimento de software é uma atividade complexa, envolvendo inúmeros fatores que, não raro, são imprevisíveis e de difícil controle, como inovações tecnológicas, imaturidade nos processos, mudanças constantes nos requisitos do cliente, dentre muitos outros. Esta complexidade faz com que grande parte dos projetos de desenvolvimento de software exceda o prazo e o orçamento previstos, além de não atender às expectativas do cliente em termos de funcionalidade e qualidade. Diante deste cenário, um gerenciamento eficaz tem-se evidenciado como de fundamental relevância para o sucesso de projetos de software. Uma vez que a incerteza é inerente a este tipo de projeto, o gerenciamento de riscos vem-se tornando cada vez mais importante neste contexto.

O gerenciamento de riscos trabalha justamente com a incerteza, visando à identificação de problemas potenciais e de oportunidades antes que ocorram, com o objetivo de eliminar ou reduzir a probabilidade de ocorrência e o impacto de eventos negativos para os objetivos do projeto, além de potencializar os efeitos da ocorrência de eventos positivos.

O gerenciamento de riscos é abordado por vários modelos dentre os quais o PMBOK (2004), o CMMI-SW (2002), a AS/NZS 4360 (2004), o padrão IEEE STD 1540-2001 (2001) e o RUP (2003). O PMBOK (A Guide to the Project Management Body of Knowledge) trata do gerenciamento de projetos de uma forma ampla, não sendo específico para software. O CMMI-SW (Capability Maturity Model Integration for Software) provê um framework para a implantação e melhoria do processo de software das

organizações. O RUP (Rational Unified Process) é um processo baseado em melhores práticas de engenharia de software [10]. A AS/NZS 4360:2004 é uma norma de âmbito mundial, específica sobre gerenciamento de riscos, desenvolvida na Austrália e Nova Zelândia.
Este trabalho analisa cada uma das abordagens de gerenciamento de riscos citadas, além de outros trabalhos relacionados, e propõe um processo para o gerenciamento de riscos em projetos de software. Esse processo poderá ser utilizado em projetos que utilizem o processo unificado ou outro processo de desenvolvimento.
Este trabalho está organizado como se segue. A seção 2 apresenta várias abordagens sobre o gerenciamento de riscos. A seção 3 propõe um processo de gerenciamento de riscos para projetos de software. A seção 4 mostra o estágio da implantação da proposta. A seção 5 apresenta as conclusões deste trabalho.

2. O GERENCIAMENTO DE RISCOS

Todo projeto tem um conjunto de riscos envolvidos e muitos deles não são descobertos até que a integração do sistema seja realizada. Riscos não identificados significam que se pode estar investindo em uma arquitetura falha ou em um conjunto não otimizado de requisitos. Além disto, a totalidade dos riscos envolvidos em um projeto está diretamente relacionada à diferença entre a estimativa mais alta de tempo de duração do projeto e a mais baixa. Para se obter estimativas acuradas é necessário identificar e tratar os riscos antecipadamente.
A seguir serão apresentadas as várias abordagens sobre o gerenciamento de riscos que foram consideradas para elaboração deste trabalho.

2.1 O Gerenciamento de Riscos no PMBOK

O PMBOK (2004) descreve o conhecimento e as melhores práticas em gerenciamento de projetos, através de nove áreas de conhecimento: Gerência de Integração, Gerência de Escopo, Gerência de Tempo, Gerência de Custo, Gerência de Qualidade, Gerência de Recursos Humanos, Gerência de Comunicação, Gerência de Riscos e Gerência de Aquisições.
A gerência de riscos inclui os processos referentes ao planejamento da gerência de riscos, à identificação, à análise, ao planejamento das respostas e à monitoração e controle dos riscos em um projeto. Esses processos interagem entre si e com os processos das outras áreas de conhecimento.
Os objetivos da gerência de risco são aumentar a probabilidade de ocorrência e os impactos de eventos positivos; e diminuir a probabilidade e os impactos dos eventos adversos aos objetivos do projeto. Os processos da gerência de risco são [9]:

- *Planejamento da gerência de riscos*: planejamento das atividades de gerenciamento de riscos a serem realizadas para o projeto.
- *Identificação de risco*: identificação dos riscos que podem afetar o projeto e documentação das suas características.
- *Análise qualitativa dos riscos*: análise qualitativa dos riscos e priorização de acordo com seus efeitos no projeto.
- *Análise quantitativa dos riscos*: mensuração da probabilidade de ocorrência dos riscos e das suas conseqüências, estimando suas implicações no projeto.
- *Planejamento das respostas aos riscos*: desenvolvimento de opções e ações para aumentar as oportunidades e reduzir as ameaças aos objetivos do projeto.
- *Monitoração e controle dos riscos*: acompanhamento dos riscos identificados, monitoramento dos riscos residuais, identificação de novos riscos, execução dos planos de respostas e avaliação da sua eficácia.

2.2 Gerenciamento de Riscos no CMMI-SW

O CMMI-SW (2002) foi criado para integrar alguns modelos do CMM, que atendem às atividades relacionadas ao desenvolvimento de software, incorporando melhorias, e ainda torná-lo compatível com a ISO/IEC 15504 (2003) [5].
O CMMI-SW contém duas representações: (i) por estágios, e (ii) contínua. A representação por estágios trata do nível de maturidade da organização como um todo, contendo cinco níveis de maturidade: inicial, gerenciado, definido, gerenciado quantitativamente e em otimização. Cada nível é constituído por um conjunto de áreas de processos, compostas por objetivos específicos (SG) e objetivos genéricos (GG). Cada objetivo específico pode ser composto por um conjunto de práticas específicas (SP).
Um objetivo específico descreve as características que devem estar presentes a fim de satisfazer uma área de processo. Uma prática específica é a descrição de uma atividade que é considerada importante para se alcançar o objetivo específico a ela associado.
A problemática do risco é abordada em três áreas de processo: Planejamento do Projeto, Monitoração e Controle do Projeto, e Gerência de Riscos. No Planejamento do Projeto, tem-se o (SG) Desenvolvimento do Plano do Projeto com a (SP) Identificar os Riscos do Projeto, que consiste na identificação e na análise dos riscos, para se determinar o impacto, a probabilidade de ocorrência e

o período em que podem ocorrer, para que os riscos possam ser priorizados. Na Monitoração e Controle do Projeto, tem-se o (SG) Monitorar o Projeto de Acordo com o Plano, onde está inserida a (SP) Monitorar os Riscos do Projeto.
A área de processo Gerência de Riscos não existia no CMM, tendo sido inserida no CMMI, o que demonstra a constatação da importância do gerenciamento de riscos e a necessidade de formalização das suas práticas para projetos de software. A Gerência de Riscos tem por finalidade identificar potenciais problemas antes que ocorram, de forma que as atividades de administração desses riscos possam ser planejadas e realizadas ao longo do ciclo de vida do produto ou projeto, para mitigar possíveis impactos adversos ao alcance dos objetivos. A Tabela 1 apresenta o relacionamento dos objetivos específicos com suas respectivas práticas específicas para essa área de processo.

Tabela 1. Relacionamento das SG e SP da Gerência de Risco do CMMI-SW (2002)

SG 1	Preparar-se para a Gerência de Riscos	
	SP 1.1	Determinar Fontes e Categorias de Riscos
	SP 1.2	Definir Parâmetros para os riscos
	SP 1.3	Estabelecer Estratégia para a Gerência de Risco
SG 2	Identificar e Analisar Riscos	
	SP 2.1	Identificar Riscos
	SP 2.2	Avaliar, Categorizar, e Priorizar Riscos
SG 3	Mitigar Riscos	
	SP 3.1	Desenvolver Planos de Mitigação de Riscos
	SP 3.2	Implementar Planos de Mitigação de Riscos

As áreas de processo Planejamento do Projeto, e Monitoração e Controle do Projeto, nível 2, tratam o gerenciamento de riscos de forma reativa, focando simplesmente na identificação dos riscos para a conscientização, e reação à medida que estes ocorram. A área de processo Gerência de Risco, nível 3, trata o gerenciamento de riscos de uma forma proativa, descrevendo a evolução das práticas específicas para sistematicamente planejar, antecipar e mitigar riscos, minimizando proativamente seu impacto no projeto.

2.3 Gerenciamento de Riscos no padrão AS/NZS 4360

O *Australian and New Zealand Standard for Risk Management* 4360 (Figura 1) é um padrão internacional para o gerenciamento de riscos, que fornece diretrizes para a estruturação e implementação de uma estratégia efetiva para o gerenciamento de riscos [1]. Enfatiza que esta estruturação varia de acordo com as necessidades da organização, seus objetivos específicos, seus produtos e serviços, bem como as práticas e os processos por ela utilizados.

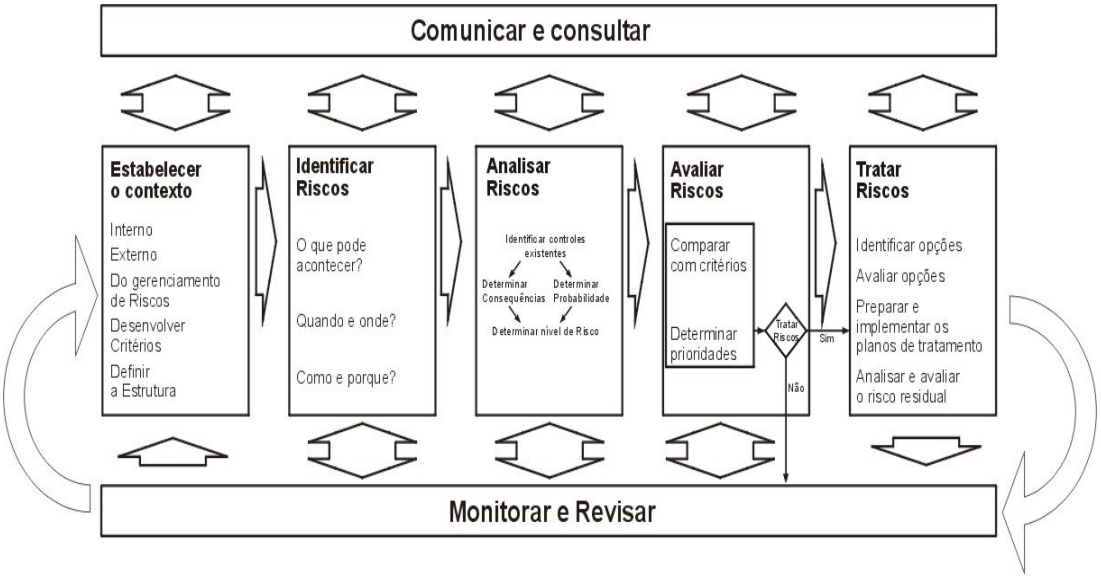


Figura 1. AS/NZS 4360:2004 Processo de Gerenciamento de Riscos

A finalidade deste padrão é estabelecer um *framework* genérico para o estabelecimento do contexto da gerência de riscos e para a identificação, a análise, a avaliação, o tratamento, a monitoração e a comunicação dos riscos na organização. A norma dá ênfase à importância de se difundir a cultura de gerenciamento de riscos por toda a organização e seus processos e pode ser aplicada em qualquer organização, independentemente do tipo de indústria ou setor econômico. Os processos de gerenciamento de risco citados são:

- *Comunicar e Consultar*: facilitar a comunicação entre os membros internos ou externos da equipe ou negócio, em cada estágio do processo de gerenciamento de riscos, a fim de atingir o sucesso do projeto.
- *Estabelecer o Contexto:* definir o contexto onde está inserido o gerenciamento de riscos, definindo os critérios de avaliação e a estrutura de análise dos riscos.
- *Identificar riscos:* identificar os eventos incertos que envolvem o projeto e quando, onde e porque podem afetá-lo.
- *Analisar os riscos:* analisar os riscos, considerando suas fontes, conseqüências positivas e negativas, probabilidade de ocorrência e nível de risco.
- *Avaliar os Riscos:* comparar os níveis de risco com os critérios pré-estabelecidos para tomar decisões sobre a extensão e natureza do tratamento requerido e sobre as prioridades.
- *Tratar os Riscos:* desenvolver e implementar as estratégias e planos de ação para ampliar os possíveis benefícios e reduzir os prejuízos decorrentes dos riscos.
- *Monitorar e Revisar:* monitorar a efetividade de todos os passos do processo para a melhoria contínua e monitorar os riscos e a efetividade do tratamento.

2.4 O padrão IEEE para Processos de Ciclo de Vida de Software - Gerenciamento de Riscos

Este padrão define um processo contínuo de gerenciamento de riscos aplicável às disciplinas de engenharia de software e de gerenciamento (Figura 2). É escrito de modo que possa ser aplicado conjuntamente com as normas da série ISO ou independentemente [4].

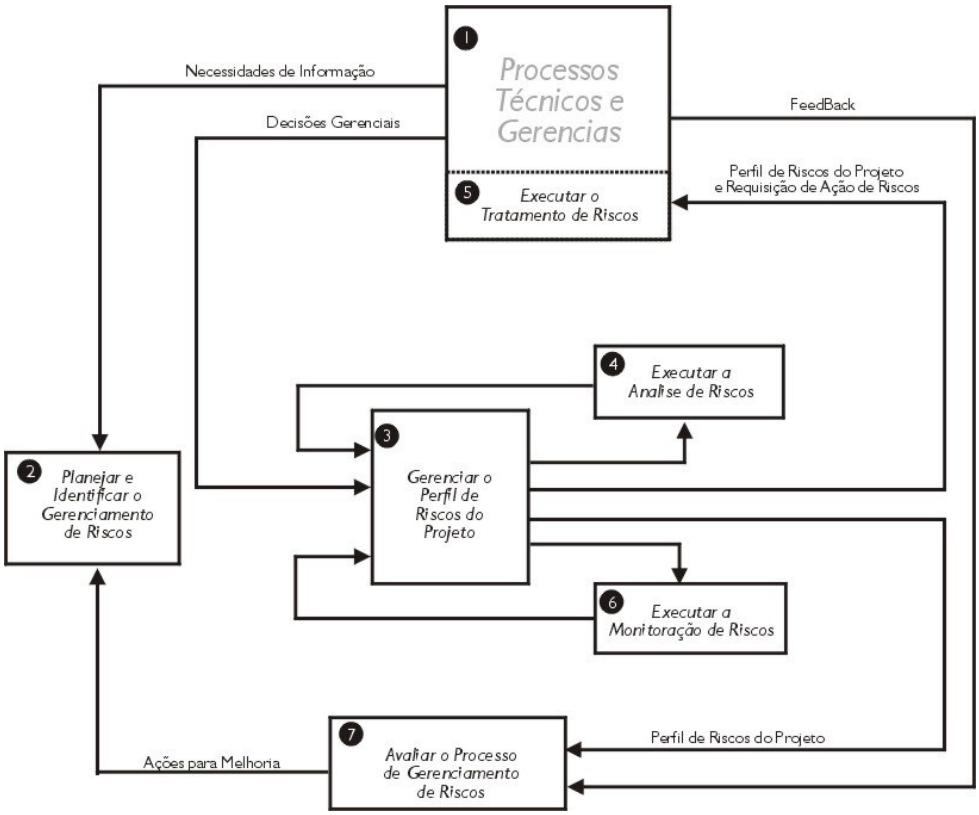


Figura 2. Modelo do Processo de Gerenciamento de Riscos – IEEE Std 1540:2001

Este processo consiste nas seguintes atividades:

- *Planejar e Implementar o Gerenciamento de Riscos:* define a política, os procedimentos e as técnicas específicas que serão utilizados para o gerenciamento de riscos; atribui os recursos e define como comunicar e coordenar os riscos e seu tratamento.
- *Gerenciar o Perfil de Risco do Projeto:* criar uma visão consistente dos riscos presentes durante o projeto e do seu tratamento, incluindo o contexto da gerência de risco, o estado atual do risco e sua história.
- *Executar a Análise do Risco:* identificar os eventos ou as situações que podem criar riscos, estimar a probabilidade de ocorrência e as conseqüências, determinar a exposição ao risco; avaliar cada risco ou combinação de riscos e gerar alternativas para tratá-los baseando-se em uma ordem de prioridade.
- *Executar o Tratamento do Risco:* determinar se os riscos são aceitáveis às partes interessadas no projeto e, caso não sejam, iniciar as ações necessárias para reduzir os riscos a um nível aceitável.
- *Executar a Monitoração do Risco:* revisar e atualizar o estado dos riscos e o contexto do gerenciamento de riscos, avaliar a eficácia do tratamento do risco e procurar novos riscos.
- *Avaliar o Processo de Gerenciamento de Riscos:* capturar informações sobre a qualidade do processo, identificar áreas para melhoria e gerar lições aprendidas.

2.5 O Gerenciamento de Riscos no RUP

O RUP [11] é um processo de engenharia de software baseado em boas práticas de desenvolvimento, especialmente no desenvolvimento iterativo, no gerenciamento de requisitos, no gerenciamento de mudanças e no direcionamento a riscos.
Segundo o RUP (2003), os riscos devem ser identificados e atacados o quanto antes no ciclo de vida do projeto, sempre objetivando a garantia da produção de software de alta qualidade, no tempo e prazo previstos e atendendo às necessidades dos usuários [8].
O RUP possui duas dimensões: a estática e a dinâmica. A primeira representa a estrutura estática do processo, descrevendo como os elementos do processo são agrupados logicamente em disciplinas. As disciplinas são agrupamentos lógicos de atividades, papéis, artefatos, conceitos e diretrizes, utilizados para a descrição de um processo, e são representadas por um fluxo de trabalho.
A dimensão dinâmica é representada pelo tempo e expressa o processo por meio de ciclos, decompostos em fases, que são divididas em iterações com marcos de conclusão. São ao todo quatro fases, que enfocam a problemática do risco de uma maneira cooperativa, conforme apresentado na Tabela 2. Um dos principais benefícios da abordagem iterativa é a identificação e o tratamento dos principais riscos do projeto em tempo hábil.

Tabela 2. Fases do RUP (2003)

Fase	Objetivos
Concepção	• Estabelecer uma boa compreensão do sistema a ser construído, por meio de requisitos de alto nível e do estabelecimento do escopo do projeto. • Obter acordo com os *stakeholders*, para prosseguir ou não com o projeto. • Preparar o ambiente para o projeto, estabelecendo o caso de desenvolvimento. • Foco no tratamento dos riscos ligados aos casos de negócio. Todos os *stakeholders* são encorajados a tomar parte. • Marco: Objetivos do Ciclo de Vida.
Elaboração	• Compreender como construir, analisando o domínio do problema. • Estabelecer uma *baseline* da arquitetura. • Desenvolver o plano de projeto, eliminando seus riscos essenciais. • Foco principalmente nos riscos técnicos, assegurando a estabilidade da arquitetura de software e, se necessário, revisando o escopo do projeto, à medida que seus requisitos tornam-se melhor compreendidos. • Marco: Arquitetura estabilizada.
Construção	• Desenvolver e testar os componentes da solução com base em critérios definidos. • Integrar os componentes da solução em um produto. • Otimizar custos, cronograma e qualidade através do gerenciamento dos recursos e do controle das operações. • Avaliar *releases* do produto. • Foco nos riscos de "logística" da construção do software, para se obter a conclusão da maior parte do trabalho. • Marco: Capacidade Operacional Inicial.
Transição	• Construir a versão final do produto, disponibilizando-a para os usuários finais. • Desenvolver artefatos de suporte à instalação e uso pelo usuário final. • Foco nos riscos associados com a logística de entrega do produto a seu usuário. • Marco: Entrega do Produto.

Cada iteração do RUP incorpora um conjunto de atividades em modelagem de negócios, requisitos, análise e projeto, implementação, testes e implantação, em várias proporções, dependendo de onde a iteração esteja localizada no ciclo de desenvolvimento. Este conjunto de atividades de engenharia de software é apoiado por três disciplinas: Gerenciamento de Projetos, Gerenciamento de Configuração e Mudanças, e Ambiente.

O gerenciamento de riscos no RUP é tratado na disciplina de Gerenciamento de Projeto, que se propõe a balancear objetivos concorrentes, gerenciar riscos e restrições, para que a entrega do produto satisfaça seus clientes e usuários. Essa disciplina provê um *framework* onde o projeto é criado e gerenciado.

O gerenciamento de riscos está integrado ao processo de desenvolvimento, onde as iterações são planejadas com base nos riscos de maior prioridade. Em uma abordagem iterativa, os riscos são mitigados mais cedo, porque os elementos são integrados progressivamente. Além disto, como cada iteração exercita muitos aspectos do projeto, torna-se mais fácil descobrir até que ponto os riscos percebidos estão se materializando, como também descobrir riscos novos [6].

O papel envolvido com o gerenciamento de riscos no RUP é o do gerente do projeto, que executa as atividades Desenvolver o Plano de Gerenciamento de Riscos, Identificar e Avaliar Riscos, e Monitorar o Status do Projeto. A seguir será descrita a proposta do processo de gerenciamento de riscos para projetos de software.

3. O PROCESSO DE GERENCIAMENTO DE RISCOS PARA PROJETOS DE SOFTWARE

O processo de gerenciamento de riscos proposto baseia-se na área de conhecimento Gerenciamento de Riscos do PMBOK (2004), na área de processo Gerenciamento de Riscos do CMMI-SW (2002), no padrão AS/NZS 4360:2004, no padrão IEEE Std 1540-2001, na abordagem de riscos do RUP (2003) e em boas práticas de gerenciamento de riscos da literatura [3] e [7].

Este processo visa auxiliar no planejamento, identificação, análise, tratamento e monitoração sistemática dos riscos em projetos de software, contribuindo para a tomada de decisão durante o ciclo de vida do projeto. Por ter sido baseado em relevantes modelos que tratam do gerenciamento de riscos, espera-se que o processo proposto possa ser utilizado em projetos de software independentemente de seu porte. Para projetos de pequeno porte, pode ser executado apenas um subconjunto das atividades do processo proposto, o que deve ser previamente definido no planejamento do gerenciamento de riscos do projeto.

O fluxo do Processo de gerenciamento de riscos para projetos de software é apresentado na Figura 3, sendo constituído dos seguintes grupamentos de atividades: (i) Planejar o Gerenciamento de Riscos, (ii) Identificar Riscos, (iii) Analisar Riscos, (iv) Tratar Riscos (v) e Monitorar e Controlar Riscos. O conjunto de atividades referentes aos itens (ii), (iii), (iv) e (v) deve ocorrer para cada iteração ao longo do processo de desenvolvimento, quando for utilizado o modelo de ciclo de vida iterativo incremental.

Para o Processo de gerenciamento de riscos proposto foram identificados os seguintes papéis como importantes para a realização de suas atividades: o Gerente do Projeto, o Especialista em Riscos e o Participante do Projeto.

O Gerente do Projeto é responsável pelo gerenciamento do projeto como um todo e participa ou é responsável por grande parte das atividades de gerenciamentos de riscos. É importante que o gerente incentive a equipe a gerenciar os riscos efetivamente.

O papel do Especialista em Riscos deve ser desempenhado por um profissional com embasamento em gerenciamento de riscos e experiência consolidada em projetos de desenvolvimento de software. O Especialista em Riscos participa de algumas atividades deste processo, algumas vezes como ator principal e, em outras, como apoio ao gerente de projeto.

O papel do Participante do Projeto pode ser desempenhado por qualquer indivíduo que possa ser afetado pelo resultado do projeto (*stakeholder*). O participante pode ser externo ou interno; o primeiro é parte interessada no projeto, e o segundo faz parte da equipe do projeto. Este último poderá ser designado como proprietário de um determinado risco, ou seja, o responsável pelas respostas ao risco planejadas [9].

O fluxo Planejar o Gerenciamento de Riscos contém as atividades detalhadas na Figura 4, que devem ser executadas no início do planejamento do projeto. Quando ocorrerem mudanças significativas, que exijam um novo planejamento do gerenciamento de riscos no projeto, este fluxo deve ser novamente executado. O objetivo deste planejamento é descrever como os elementos e recursos do processo de gerenciamento de riscos serão tratados no projeto. Nesse fluxo, o especialista em riscos determina as fontes e as categorias de riscos com base nos ativos do processo organizacional e nas características do projeto a ser desenvolvido. Define também, os parâmetros a serem utilizados para analisar e categorizar riscos, e para controlar o esforço da gerência de risco.

Os riscos do projeto podem ser categorizados por área do projeto afetada, por fase do de ciclo de vida do projeto ou outra categorização que seja útil para determinar as áreas do projeto mais expostas aos efeitos da incerteza [9]. Em seguida, o gerente de projeto estabelece a estratégia a ser adotada, para a condução das atividades de gerenciamento de riscos no projeto. Esta estratégia envolve a metodologia a ser utilizada, a atribuição dos papéis e responsabilidades, a definição da periodicidade da monitoração e controle dos riscos, o planejamento de como se dará a comunicação interna e externamente no projeto e a definição da avaliação do processo de gerenciamento de riscos. Estas informações devem ser registradas no Plano de Gerenciamento de Riscos, que descreve como o gerenciamento de riscos deverá ser estruturado e executado no projeto.

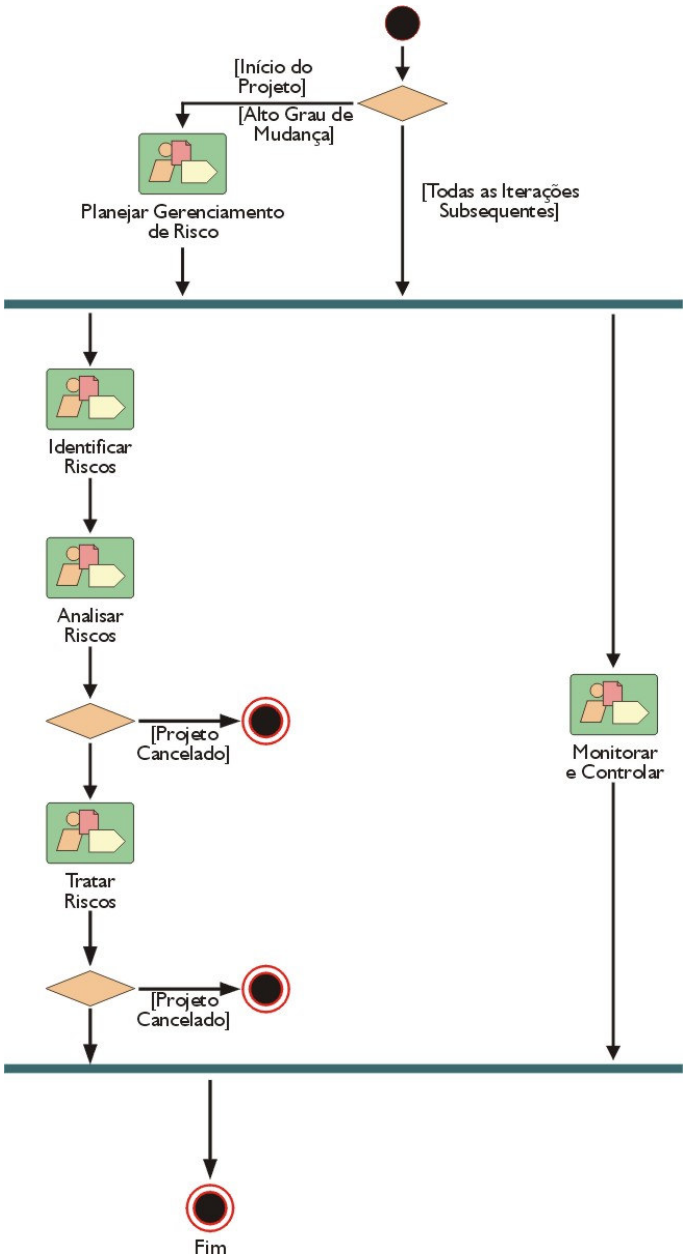


Figura 3. Fluxo do Processo Gerenciamento de Riscos

A Figura 5 detalha as atividades do fluxo Identificar Riscos, que consiste em descobrir os riscos que podem ocorrer em um projeto específico e estabelecer o seu contexto, verificando quais deles podem afetar os objetivos do projeto.

A existência de informação de boa qualidade é fundamental para a identificação de riscos, assim como dados históricos sobre projetos similares e a subseqüente discussão com os participantes do projeto. O envolvimento de participantes internos e externos também ajuda no compromisso para com o processo de gerenciamento de riscos e a garantir que os riscos tenham o mesmo significado para os diferentes participantes [1].

A identificação dos riscos deve ser feita de uma forma organizada e completa a fim de identificar riscos mais prováveis e cuja ocorrência irá afetar os objetivos do projeto. Para ser eficaz, a identificação dos riscos não deve ser uma tentativa de referir-se a cada evento possível, sem levar e conta o quão improvável ele possa ser. O uso das fontes, categorias e parâmetros desenvolvidos no planejamento do gerenciamento de riscos podem fornecer a disciplina e a otimização apropriadas à identificação do risco [2].

As características dos riscos devem ser documentadas em uma linguagem concisa, incluindo o contexto, as condições e as conseqüências de sua ocorrência. Essas atividades geram o artefato Registro do Risco cuja preparação começa com a identificação dos riscos e tem continuidade com a análise, o tratamento e a monitoração e controle dos riscos.

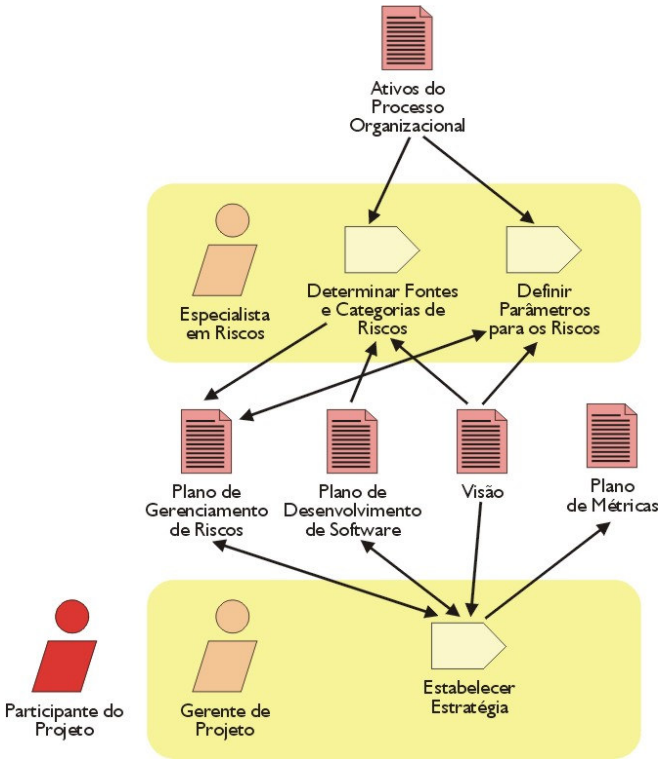


Figura 4. Fluxo Planejar o Gerenciamento de Riscos

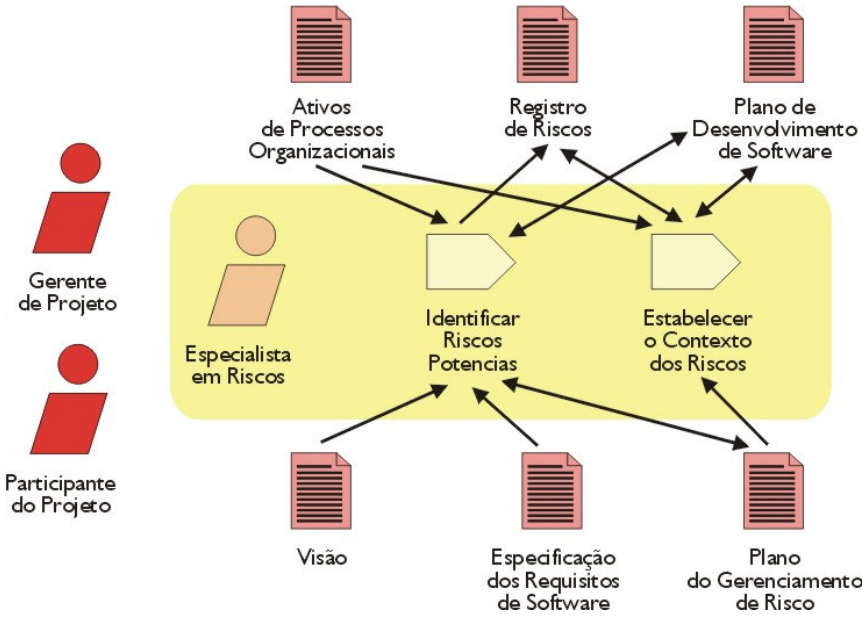


Figura 5. Fluxo Identificar Riscos

A Figura 6 detalha as atividades do Fluxo Analisar Riscos, que consiste tipicamente na categorização, avaliação e priorização dos riscos. A análise dos riscos tem por objetivo estabelecer uma compreensão do nível do risco e da sua natureza. Os riscos similares devem ser agrupados de acordo com as categorias selecionadas no passo Determinar as Categorias de Riscos do fluxo Planejar o Gerenciamento de Riscos. A categorização dos riscos pode conduzir ao desenvolvimento de respostas mais eficazes aos riscos.

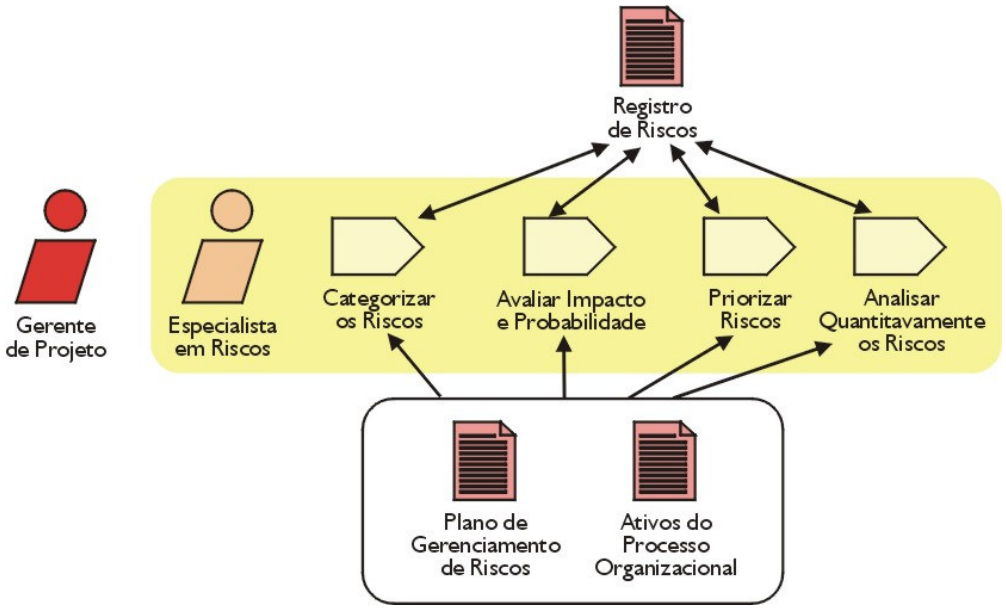


Figura 6. Fluxo Analisar Riscos

A avaliação do impacto e da probabilidade de ocorrência dos riscos é necessária para determinar a importância relativa de cada risco identificado para efeito de priorização, e é usada também para chamar a atenção da gerência. Os riscos devem ser priorizados de acordo com a exposição geral que eles representam para o projeto e com os critérios estabelecidos. O objetivo da priorização é determinar as áreas onde os recursos devem ser aplicados com um maior impacto positivo para o projeto.

A forma de expressar o impacto, a probabilidade e a combinação de ambos para fornecer o nível de risco, variará de acordo com o tipo de risco e o objetivo da avaliação. A incerteza e a variação do impacto e da probabilidade devem ser consideradas na análise e comunicadas eficazmente.

De acordo com a norma AS/NZS 4360:2004, o gerenciamento de riscos pode ser usado para identificar e priorizar os riscos positivos ou oportunidades, o que muda é apenas o foco na ação de capturar e explorar a oportunidade.

A Figura 7 detalha as atividades do fluxo Tratar Riscos, que consiste na definição de estratégias para lidar com as ameaças e as oportunidades e na definição e seleção das ações para implementá-las.

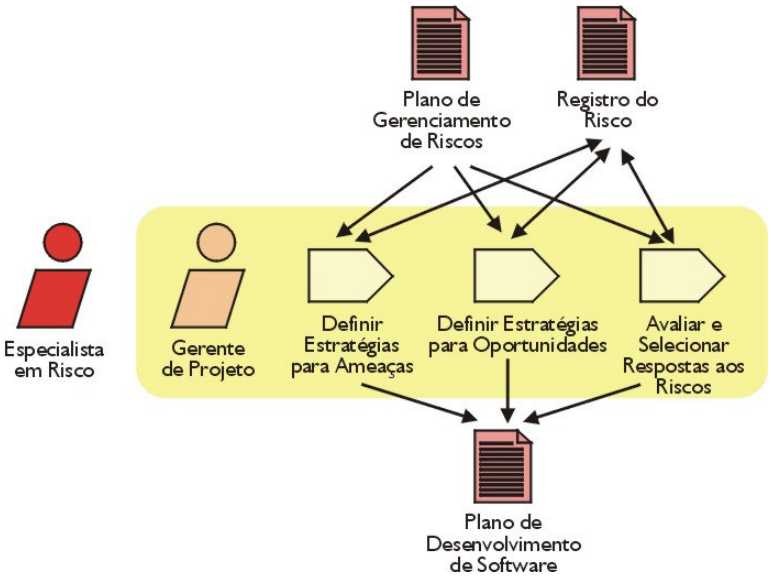


Figura 7. Fluxo Tratar Riscos

A estratégia ou combinação de estratégias mais apropriada para tratar os riscos deve ser selecionada levando-se em consideração alguns fatores, entre eles a relação custo/benefício.

As estratégias para responder às ameaças incluem evitar, transferir, aceitar ou mitigar o risco. A mitigação consiste em tomar uma ação imediata e proativa para reduzir a probabilidade e/ou o impacto do risco.
As estratégias para potencializar as oportunidades consistem em explorar, compartilhar, elevar a categoria ou aceitar o risco.
As estratégias de contingência referem-se a desenvolver planos alternativos, que serão postos em prática no caso do risco se tornar realidade, e na criação de reservas que serão incluídas no Plano de Desenvolvimento de Software, cujo objetivo é minimizar riscos relacionados a custo e prazo.
A Figura 8 detalha as atividades do Fluxo Monitorar e Controlar Riscos, que envolve os demais fluxos do Processo de Gerenciamento de Riscos, os artefatos gerados por estes fluxos devem ser revisados no decorrer do projeto. A atividade Monitorar o Status dos Riscos consiste em revisar os riscos, avaliar se os limites estão em um ponto crítico, que exija a execução do plano de contingências, comunicar o status e executar as atividades previstas para os riscos realizados.

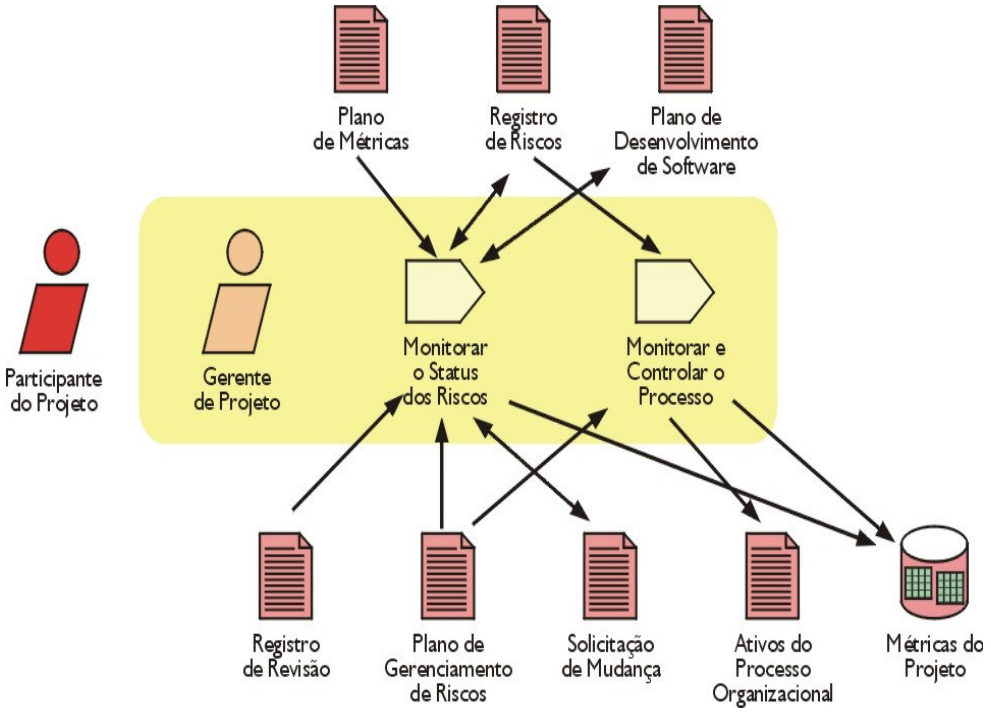


Figura 8. Fluxo Monitorar e Controlar Riscos

Dependendo do grau de mudanças exigido pode ser necessário acionar o fluxo Planejar Gerenciamento de Riscos, pois pode indicar a necessidade de um novo planejamento. A atividade Monitorar e Controlar o Processo consiste em verificar a eficácia do processo através dos artefatos produzidos, medidas e resultados sobre a melhoria derivada do planejamento e execução do processo de gerenciamento de riscos. Estas informações irão suportar o uso e a melhoria do processo de software da organização e dos recursos do processo.
A comunicação é fundamental durante todo o processo para garantir que as pessoas responsáveis por implementar o gerenciamento de riscos, e também as partes interessadas no projeto, entendam as bases nas quais as decisões serão tomadas, e o porquê de determinadas ações serem adotadas.
Cada uma das atividades de cada um dos fluxos constantes do Processo de Gerenciamento de Riscos está detalhada em termos de propósitos, passos, artefatos de entrada e de saída e atores envolvidos. A Tabela 3 contém cada fluxo de trabalho, com suas respectivas atividades e passos.

4. Implementação do Processo de Gerenciamento de Riscos

Este trabalho está em processo de validação em uma empresa de grande porte, que utiliza o framework RUP em seu processo de desenvolvimento de software. Essa empresa possui um conjunto de sistemas legados, desenvolvidos na abordagem da análise estruturada. Os novos sistemas estão sendo desenvolvidos no paradigma da orientação a objetos.

Inicialmente, foram realizados treinamentos na empresa, abordando o processo de desenvolvimento do RUP, tendo sido destacada a importância do gerenciamento de riscos, para o sucesso dos projetos de desenvolvimento, no contexto da organização. A disseminação do conhecimento sobre o processo de gerenciamento de riscos de software proposto foi feita para um conjunto de seus funcionários.

Tabela 3. Detalhamento dos Fluxos e Atividades

Detalhamento de Fluxo de Trabalho	Atividades	Passos
Planejar o Gerenciamento de Riscos	Determinar Fontes e Categorias dos Riscos	Determinar as Fontes de Riscos
		Determinar as Categorias de Riscos
	Definir Parâmetros para os Riscos	Definir Critérios para Comparação dos Riscos
		Definir Indicadores
	Estabelecer Estratégia	Definir Metodologia para o Gerenciamento de Riscos
		Atribuir Papéis e Responsabilidades
		Definir a Periodicidade da Monitoração e Controle dos Riscos
		Planejar a Comunicação
		Definir a Avaliação do Processo de Gerenciamento dos Riscos
Identificar Riscos	Identificar Riscos Potenciais	Analisar a Documentação do Projeto
		Identificar Riscos
	Estabelecer o Contexto dos Riscos	Detectar as circunstâncias ou condições do risco
		Determinar quando o risco pode ocorrer
		Identificar Possíveis Conseqüências
Analisar Riscos	Categorizar os Riscos	Agrupar Riscos de Acordo com as Categorias
	Avaliar Impacto e Probabilidade	Avaliar o Impacto dos Riscos para os Objetivos do Projeto
		Avaliar a Probabilidade de Ocorrência dos Riscos
		Determinar a Exposição ao Risco
	Priorizar Riscos	Determinar a Prioridade Relativa dos Riscos
	Analisar Riscos Quantitativamente	Analisar Riscos Quantitativamente
Tratar Riscos	Definir Estratégias para Ameaças	Definir Ações para Evitar Riscos
		Definir Ações para Transferir Riscos
		Definir Ações para Mitigar Riscos
	Definir Estratégias para Oportunidades	Definir Ações para Explorar Riscos
		Definir Ações para Compartilhar Riscos
		Definir Ações para Melhorar os Riscos
	Avaliar e Selecionar Respostas aos Riscos	Avaliar e Selecionar as Opções de Tratamento
		Definir os Riscos que Serão Aceitos
		Definir Plano de Contingência
Monitorar e Controlar Riscos	Monitorar o Status dos Riscos	Revisar Riscos
		Avaliar Status dos Riscos
		Monitorar o Tratamento dos Riscos
		Comunicar Status dos Riscos
	Monitorar e Controlar o Processo	Verificar Conformidade com o Plano
		Coletar Informações sobre o Processo
		Avaliar e Melhorar o Processo

Foram realizadas também algumas entrevistas com gerentes de projetos, a fim de levantar dados sobre projetos importantes realizados na organização, com o objetivo de obter subsídios para execução do fluxo Planejar o Gerenciamento de Riscos. Através dessas entrevistas, foi elaborado um documento preliminar referente ao seguinte artefato: *Ativos de Processos Organizacionais*.

Como resultado dessas entrevistas, foi verificado que as atividades referentes ao gerenciamento de riscos ainda eram muito insipientes na organização. Isto se deve ao fato de que o gerenciamento de riscos não faz parte da cultura organizacional, evidenciando a necessidade de um processo específico para estas atividades.

Em seguida, foram selecionados dois projetos pilotos na organização, para se aplicar o processo de gerenciamento de riscos proposto. Os critérios utilizados na seleção foram a importância dos projetos para os negócios da empresa, e o nível de riscos envolvidos nos mesmos.

O primeiro desses projetos consiste na migração de é um sistema de grande porte e em mainframe, que utiliza a abordagem estruturada, para a plataforma *Web*, onde serão desenvolvidas novas funcionalidades. Como este projeto já havia sido iniciado antes de ter sido aplicado o processo de gerenciamento de riscos proposto, percebeu-se que alguns dos problemas ocorridos, poderiam ter sido evitados como, por exemplo:

- O escopo do projeto não foi bem definido;

- A maioria dos requisitos do sistema não está documentada, sendo este conhecimento dominado apenas por algumas pessoas da equipe do sistema legado;
- A questão da integração da base de dados do sistema legado com a base de dados do novo sistema não foi atacada convenientemente, isto é, ficaram muitas pendências para serem resolvidas a posteriori, não havendo uma priorização adequada. Neste caso, os riscos não foram totalmente identificados, analisados e nem tratados adequadamente, isto é, não foram definidas estratégias de mitigação e nem desenvolvidos planos de contingência.

Este projeto ainda está em andamento, mas já é possível constatar alguns benefícios com a utilização do processo de gerenciamento de riscos de software proposto, a partir da iteração em que o mesmo foi inserido.
Para o segundo projeto, que está sendo desenvolvido na abordagem de orientação a objetos, a partir do framework do RUP, a aplicação do processo de gerenciamento de riscos está em sua fase inicial, não havendo ainda resultados concretos. Espera-se que para este segundo projeto, o processo proposto seja mais facilmente aplicado, uma vez que o RUP já tem a preocupação com o gerenciamento de riscos.
Este processo iniciou sua utilização em uma segunda empresa também de grande porte, certificada CMM nível 2 e que está buscando a certificação CMMI nível 3, para cobrir a área de processo de Gerenciamento de Riscos. Foi feita a especialização do processo para a empresa e está sendo iniciada a sua utilização em quatro projetos piloto. Entretanto, ainda não se tem resultados conclusivos sobre essa segunda empresa.

5. CONCLUSÃO

O processo de gerenciamento de riscos para projetos de software, proposto neste trabalho, traz as seguintes contribuições:

- Destacar a importância da gerência de riscos para projetos de software, agregando em um só processo, os fundamentos de modelos de riscos reconhecidos pela comunidade científica, e que são amplamente utilizados;
- Formalizar um processo de gerenciamento de riscos para projetos de software em geral. Esse processo é especialmente aderente ao ciclo de vida iterativo incremental, no qual o RUP se enquadra;
- Contribuir para que o gerenciamento de riscos em projetos de software seja mais efetivo, resultando em um desenvolvimento de software com qualidade;
- Atender à Área de Processo Gerência de Riscos do CMMI-SW, nível 3, e às exigências do PMBOK (2004);
- Fornecer um guia para a implantação da área de processo Gerência de Riscos do CMMI-SW, nível 3;
- Enfatizar a utilização dos ativos do processo organizacional no gerenciamento de riscos para projetos específicos;
- Fazer uso do conceito de riscos positivos ou oportunidades, assim como as estratégias para potencializar seus efeitos para projetos de desenvolvimento de software.

Referências

[1] AS/NZS 4636 (2004), Australian/New Zealand Standard for Risk Management.

[2] CMMI Product Team (2002), CMMI for Systems Engineering/Software Engineering, Version 1.1 Staged Representation (CMU/SEI-2002-TR-029, ESC-TR-2002-029), Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

[3] Gusmão, C. M. G., Moura, P. H., (2003), ISO, CMMI, and PMBOK Risk Management: a Comparative Analysis, The International Journal of Applied Management and Technology, Volume 1, Number 1.

[4] IEEE Std 1540-2001 (2001), Standard for Software Life Cycle Processes-Risk Management.

[5] ISO/IEC 15504, (2003), Software Process Assessment.

[6] Kroll, Per & Kruchten, Philippe, (2003), The Rational Unified Process made easy: a practitioner's guide to the RUP, Pearson Education.

[7] Machado, C. A. F., (2002), A-Risk: Um Método para Identificar e Quantificar Risco de Prazo de Projetos de Desenvolvimento de Software, Dissertação de Mestrado, PUC-PR, Curitiba.

[8] Manzoni, Lisandra V. & Price, Roberto T., (2003), Identifying Extensions required by RUP (Rational Unified Process) to comply with CMM (Capability Maturity Model) Levels 2 and 3, IEEE Transactions on Software Engineering, Vol. 29. No. 2, February.

[9] PMBOK (2004), PMI Standards Committee. A Guide to the Project Management Body of Knowledge, Third Edition, PMI Publishing Division, Philadelphia, USA.

[10] Reinehr, S.S., Balduino, R., Machado, C. A. F., Pessoa, M. S., (2003), Implementing ISO/IEC 12207 Standard using Rational Unified Process. Software Engineering Research and Practice.

[11] RUP (2003), Rational Unified Process, Version 2003.06.00.65, CD-ROM. Rational Software Corporation, Cupertino, California.

Mejoramiento del Proceso de Pruebas en un Contexto de Desarrollo de Software Globalizado

Rubby Casallas, Darío Correal, Nicolás López
Universidad de Los Andes, Departamento de Sistemas y Computación,
Bogotá D.C., Colombia
{rcasalla, dcorreal, ni-lopez}@uniandes.edu.co

Abstract

Development groups in a global context face great challenges. These are related with problems in communication, work synchronization, project administration, scheduling, etc. To reduce the problems, it is necessary that the involved people have a clear understanding of the used software processes to carry out their tasks, and they should have available a proper technological infrastructure to support these processes and ease coordination. In this paper, we present a proposal for the improvement of a testing software process in a global software development context. The proposal covers the definition of the process and the infrastructure to support the enactment in a geographically disperse context. The infrastructure is based on distributed events where coordination between applications is achieved by means of Event-Condition-Action rules.

Keywords: Software Engineering, Global Software Development, Software Testing, Event Middleware, Application Integration

Resumen

Los grupos de desarrollo de software en un contexto globalizado enfrentan grandes retos. Estos están relacionados con problemas de comunicación, sincronización del trabajo, administración del proyecto, horarios, etc. Para aliviar parte de los problemas es necesario que las personas involucradas tengan claros los procesos de software que utilizan para llevar a cabo sus tareas y que cuenten con una infraestructura tecnológica que apoye estos procesos y facilite la coordinación. En este artículo presentamos una propuesta para el mejoramiento del proceso de pruebas y corrección de defectos en un ambiente de desarrollo de software globalizado. La propuesta consta de la definición del proceso y de la infraestructura que dará soporte a la ejecución en el contexto de dispersión geográfica. La infraestructura está basada en eventos distribuidos donde la coordinación entre aplicaciones se logra por medio de reglas Evento-Condición-Acción[1].

Palabras claves: Ingeniería de Software, Desarrollo de Software Globalizado, Pruebas de Software, Plataforma de Eventos, Integración de Aplicaciones

1 INTRODUCCIÓN

El mejoramiento continuo de los procesos de desarrollo de software es cada día más importante para las empresas de desarrollo que pretenden competir abiertamente en los mercados internacionales, especialmente aquellas que cuentan con equipos de trabajo ubicados en diferentes puntos geográficos.

Actualmente se utilizan diferentes metodologías y estándares de certificación de los procesos de desarrollo de software, tales como: TSP[19], RUP[20], XP[11] y CMMi[6] entre otros. Sin importar cual metodología sea utilizada por un equipo de desarrollo, las pruebas realizadas sobre el producto, ya sean unitarias, de integración o de sistema, reflejan si los objetivos planteados a nivel de requerimientos son satisfechos por el producto desarrollado. Este proceso, típicamente involucra el diseño e implementación de las pruebas, su ejecución, el reporte de los defectos encontrados, la planeación de las correcciones y la implementación de dichas correcciones. Estos pasos no siempre están formalmente definidos y en muchas ocasiones se encuentran en las cabezas de los miembros del equipo de trabajo quienes deciden en un momento dado cómo manejar la información asociada a las pruebas y qué pasos seguir durante este proceso.

[1] Esta propuesta está siendo validada en un proyecto con una casa de desarrollo de software en Colombia, Heinsohn Software House S.A., que enfrenta los retos de desarrollo globalizado. El proyecto se desarrolla con el auspicio y financiación parcial del Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología "Francisco José de Caldas" - COLCIENCIAS.

Esta falta de manejo de conocimiento y de mecanismos automatizados de coordinación entre los miembros de un equipo de trabajo pueden ser catastróficos para un proceso de desarrollo de software[25][26], generalmente este problema es solucionado en algún porcentaje mediante la interacción de los miembros de un equipo de trabajo. Esta interacción está apoyada por medios de comunicación tales como reuniones presenciales, tele-conferencias, correos electrónicos y en general otras formas de intercambio de conocimiento verbal. Este mismo problema, ahora visto desde el punto de vista de equipos de trabajo geográficamente dispersos, aumenta en complejidad y en riesgo[9]. De una parte los miembros de trabajo en un ambiente de desarrollo de software globalizado (GSD por sus siglas en inglés) no se conocen, probablemente hablan idiomas diferentes, trabajan en horarios diferentes y muy seguramente carecen de los espacios comunes para un intercambio de información verbal en la solución y coordinación de un proceso de pruebas[27][31].

Este artículo tiene dos objetivos principales. El primero es presentar la definición de un proceso de pruebas y manejo de defectos teniendo en cuenta el contexto de dispersión geográfica de los desarrolladores del proyecto: las actividades que se deben realizar y la sincronización del trabajo. El segundo objetivo es presentar la infraestructura de soporte diseñada para la implementación de dicho proceso. La infraestructura consiste por un lado de las aplicaciones que dan apoyo a las actividades asociadas al proceso de pruebas y por otro lado, de un mecanismo de integración de esas aplicaciones, basado en una plataforma de eventos y en reglas Evento-Condición-Acción[5].

La organización del resto de este documento es la siguiente. En la sección dos se presentan las limitaciones encontradas en un proceso de pruebas GSD. En la sección tres se presenta el proceso propuesto. En la sección cuatro se presentan las aplicaciones y la infraestructura de integración. En la sección cinco se presentan los resultados obtenidos en la utilización de este proceso. En la sección seis se presentan los trabajos relacionados, y por último en la sección siete se presentan las conclusiones y trabajos futuros.

2 LIMITACIONES DE UN PROCESO DE PRUEBAS GSD

En la actualidad, cada día es más usual ver como los equipos de desarrollo de software requieren trabajar de forma distribuida; esta característica de trabajo, es en principio vista como una solución viable para varios problemas, tales como falta de recursos locales, fábricas de software con necesidades de producción 24 horas y recursos de desarrollo calificado a mejores precios. Si bien todos esto es cierto, no son menos ciertos los problemas técnicos y organizacionales que aparecen debido a la distribución de los miembros de un equipo de desarrollo[4][18].

Según[14] los procesos GSD pueden ser afectados a diferentes niveles y en diferentes aspectos, tales como: estratégicos, culturales, comunicativos, de procesos, técnicos y de conocimiento. En particular desde el punto de vista de la comunicación, un proceso GSD puede tener las dificultades más grandes y a la vez las más difíciles de solucionar, dado que en este tipo de procesos es donde más se siente la ausencia de interacción entre los miembros de un equipo, ya que usualmente las comunicaciones informales apoyan la labor de desarrollo.

En esta sección queremos resaltar dos tipos de problemas encontrados en equipos de desarrollo distribuidos, los cuales serán tenidos en cuenta en el momento de proponer el nuevo proceso de pruebas y las herramientas de soporte de dicho proceso, estos problemas son: problemas de coordinación de actividades y problemas de integración de las aplicaciones utilizadas por los equipos de desarrollo.

El primer tipo de problema encontrado frecuentemente en equipos de trabajo GSD es el de la coordinación de los miembros del grupo[16]. Estos problemas suelen presentarse como consecuencia de la dispersión geográfica, de las diferencias horarias y de las diferencias culturales de las personas participantes, tal como lo sugiere un estudio realizado[15] en donde se resalta éste como el factor más relevante para generar retardos y fallas en proyectos geográficamente distribuidos.

Desde el punto de vista de procesos y manejo de proyectos en [14] se resalta con especial énfasis la ausencia de sincronización entre las tareas de los miembros de un equipo de trabajo, donde frecuentemente no hay una definición clara de las entradas y salidas esperadas de una tarea, unido al hecho de que frecuentemente no se cuenta con herramientas que faciliten la comunicación y colaboración entre los miembros del equipo [16].

De otra parte, como un segundo problema está la heterogeneidad en las herramientas de trabajo utilizadas por los equipos de desarrollo [29]. Esta heterogeneidad suele presentar un problema para la coordinación de actividades, en donde es común ver cómo el proceso de pruebas y corrección de defectos involucra herramientas diversas para el registro de planes de prueba, ejecución de las pruebas, registro de defectos y planeación de la corrección. Para poder

dar soporte al proceso globalizado, estas aplicaciones deben estar integradas de manera transparente para quien está ejecutando las tareas. Es decir, se espera que cuando se registre un defecto, automáticamente, en la aplicación de planeación, se cree una nueva tarea de corrección. Esta automatización facilitará la coordinación de las personas, dado que la comunicación entre ellas se reduce y parte de ésta queda reflejada en las reglas de integración de las aplicaciones.

3 PROCESO DE PRUEBAS PROPUESTO

En esta sección se explica en detalle el proceso de pruebas propuesto para una empresa de desarrollo de software que maneja equipos de trabajo geográficamente distribuidos, posteriormente se presentan las herramientas de apoyo construidas para la ejecución del proceso y su implementación. A continuación se presentan las principales definiciones utilizadas en el proceso de pruebas, tales como: planes de prueba, pruebas, casos de prueba, ejecuciones y defectos.

Un plan de pruebas es la unidad básica de ejecución dentro del sistema de pruebas, está asociado a un proyecto de desarrollo y está constituido por un conjunto de pruebas y ejecuciones. Una prueba pertenece a un plan de pruebas e indica el tipo de prueba que se desea realizar dentro del proyecto, las pruebas están compuestas por casos de prueba. Esta relación se puede ver claramente en la figura 1

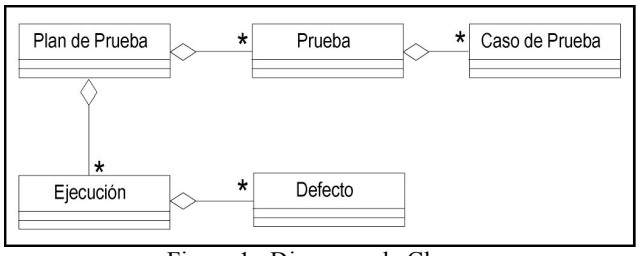


Figura 1. Diagrama de Clases

Un caso de prueba corresponde a un conjunto de datos de entrada con el que se ejecutará la prueba y un conjunto de criterios de aceptación para el caso de prueba. Una ejecución de un plan de pruebas, significa que se ejecutaron todas las pruebas y para cada una, todos sus casos de prueba definidos. El resultado de una ejecución de un plan de pruebas es un conjunto de defectos que puede ser vacío si todo funcionó correctamente, en caso contrario, cada defecto corresponde a un caso de prueba cuyo criterio de aceptación no fue satisfecho.

El proceso de pruebas propuesto está basado un modelo en V[33], es decir que en forma paralela al desarrollo del producto se van creando los planes de prueba, pruebas y casos de prueba con sus criterios de aceptación para que al término del desarrollo se tengan listas las pruebas que deben llevarse a cabo. Estas pruebas pueden ser unitarias, ejecutadas por el desarrollador o de aceptación las que usualmente involucran pruebas a todo el sistema realizadas por usuarios finales. En la Figura 2 se muestra este proceso.

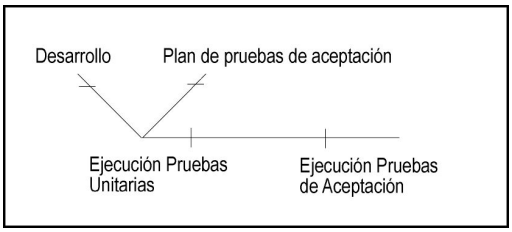


Figura 2. Modelo en V

El proceso consiste en ciclos cortos de desarrollo y pruebas[12], en donde cada ciclo tiene una duración de entre 4 a 8 semanas. En un contexto más general, el desarrollo completo del proyecto puede apreciarse en la figura 3 en la que diferentes ciclos son ejecutados uno tras otro, hasta finalizar el producto completo. De esta manera antes de comenzar cada ciclo, se determinan los requerimientos que serán desarrollados, generalmente un conjunto pequeño que pueda ser implementado en corto tiempo[1][7].

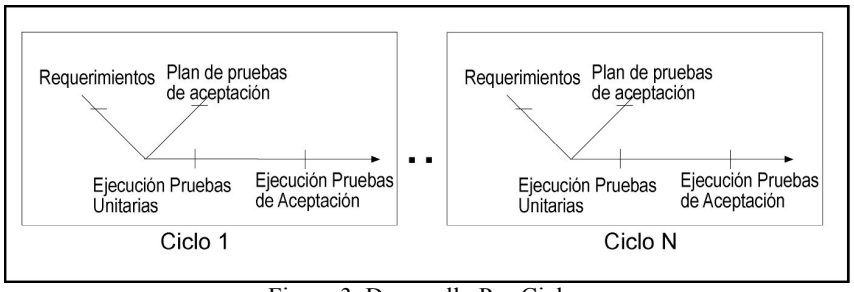


Figura 3. Desarrollo Por Ciclos

4 HERRAMIENTAS DE APOYO AL PROCESO

Como apoyo a la ejecución del proceso propuesto, se utilizan herramientas que dan soporte a las distintas actividades y una plataforma de integración. La plataforma de integración facilita la utilización de las otras aplicaciones en un ambiente de desarrollo GSD, así como de cualquier otra aplicación que requiera el equipo de trabajo, con algunas limitaciones que serán presentadas más adelante.

4.1 Aplicaciones Utilizadas en el Proceso de Pruebas

Para adelantar el proceso de pruebas propuesto en este trabajo se desarrolló una aplicación llamada Cronos y se utilizaron tres aplicaciones ya desarrolladas con anterioridad llamadas: Delfos, Arquímedes y Vernue. Dado que estas aplicaciones han sido desarrolladas en diferentes tiempos y por diferentes grupos de desarrollo ninguna tiene conocimiento de las otras. Estas aplicaciones serán explicadas a continuación.

La herramienta Delfos tiene como objetivo la administración de los requerimientos identificados en el desarrollo de un producto. Gracias a que esta herramienta puede ser utilizada vía Web, el levantamiento de requerimientos puede llevarse a cabo en las instalaciones del cliente o en varios de los sitios de desarrollo de la empresa de software sin importar su ubicación geográfica o su horario de trabajo. A medida que los requerimientos van siendo aprobados por el cliente se hacen visibles a todos los miembros del equipo y en especial a los encargados de la creación de planes de prueba, quienes pueden comenzar a ejecutar el ciclo en V, de tal forma que la diferencia horaria y geográfica no sea un impedimento.

El equipo de creación de planes de prueba comienza el desarrollo de casos de pruebas y pruebas del sistema y los registra en la herramienta Arquímedes. Esta aplicación, al igual que la herramienta Delfos, está desarrollada en un ambiente Web, lo que facilita la consulta de los planes de prueba que se desarrollan y de los criterios de satisfacción a ser utilizados para determinar si la prueba es correcta o no. Una vez comienza la ejecución de los planes de prueba, sus resultados son documentados por los miembros del equipo de pruebas y dependiendo de los resultados se notifican los defectos encontrados para generar la planeación correspondiente a la corrección de defectos y la programación de la ejecución de nuevos planes de prueba.

Las actividades que se han mencionado anteriormente no son coordinadas directamente por las personas que participan en el proyecto, ya que en muchos casos éstas trabajan en diferentes lugares y zonas horarias. Para suplir esta coordinación se desarrolló la herramienta Cronos, la cual permite crear tareas asociadas a cada una las actividades de ejecución de planes de prueba y corrección de defectos, estas tareas son asignadas automáticamente a los responsables de llevarlas a cabo, permitiendo la coordinación automática de las labores a desarrollar y un control de las actividades desarrolladas por cada miembro del equipo de trabajo. La herramienta Cronos al igual que las dos anteriores, está desarrollada sobre un ambiente Web lo que permite ser ejecutada y consultada desde cualquiera de los puntos de trabajo del equipo de desarrollo.

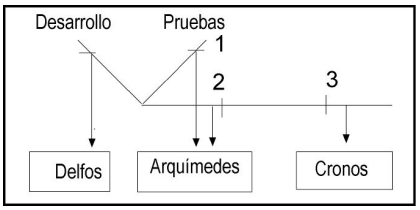


Figura 4. Aplicaciones utilizadas para el modelo en V

En la figura 4 es posible ver de forma general el proceso descrito y las aplicaciones desarrolladas. Una vez terminado el ciclo en V, es decir, una vez terminado el desarrollo de los requerimientos, al igual que los respectivos planes de prueba que los validarán, mostrado en la figura 4 con el número 1, se pasa a la siguiente etapa, mostrada en la figura 4 con el número 2. Esta etapa está asociada a la ejecución de los planes de prueba y al registro de los defectos y mejoras resultado de las diferentes ejecuciones de los casos de prueba. Cuando se termina el reporte de defectos, se comienza la etapa de planeación para la corrección, este paso es mostrado en la figura 4 con el número 3. *Cronos* le permite a un usuario conocer las actividades y tareas asignadas para la corrección de los defectos, a la vez que le facilita al responsable del proyecto conocer el estado global, el estado de las pruebas y determinar tiempos de finalización del proyecto.

Vernue es una aplicación para control de tiempos, seguimiento de proyectos y reportes estadísticos. Esta aplicación a diferencia de las anteriores herramientas no es una aplicación Web. Vernue se integra de manera transparente al proceso cuando, una vez terminada la ejecución de las tareas, en especial las de correcciones de defectos, los tiempos totales invertidos allí son registrados para producir estadísticas y facilitar el costeo de los proyectos.

Las tres primeras herramientas (Cronos, Arquímedes y Delfos) fueron desarrolladas sobre plataformas J2EE, Vernue fue desarrollada para un ambiente AS400.

4.2 Infraestructura de Apoyo a la Integración de Aplicaciones

En la sección anterior se presentó la utilización de las herramientas de soporte al proceso, sin embargo es importante mencionar que ninguna de estas aplicaciones conoce de la existencia de las otras, aunque en el proceso general están fuertemente relacionadas e interactúan entre si. Para poder llevar a cabo esta interacción conservando el desacoplamiento de las aplicaciones, se desarrolló una infraestructura distribuida de eventos llamada Eleggua. Esta infraestructura facilita la comunicación y la coordinación de aplicaciones distribuidas, independizando su ubicación geográfica y tecnología utilizadas para su implementación.

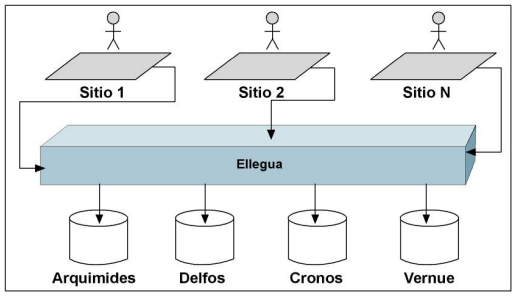


Figura 5. Desacoplamiento de aplicaciones mediante el uso de Eleggua

La funcionalidad principal de Eleggua es la de permitir la observación de eventos ocurridos y la notificación de los mismos a las aplicaciones interesadas en dichos eventos. Mediante este mecanismo, las aplicaciones geográficamente dispersas se mantienen desacopladas a la vez que participan en conjunto en el proceso de pruebas mediante la generación y consumo de eventos.

Para llevar a cabo la observación de las aplicaciones se utilizan reglas de observación, en las que se especifica la intercepción de un servicio de interés en una aplicación y se describe cómo esta intercepción debe ser procesada para generar un evento lógico. Los eventos lógicos son la pieza fundamental en el intercambio de información entre

aplicaciones y están compuestos por un tipo de evento, la identificación del productor del evento, una estampilla de tiempo y un conjunto de parámetros.

El servicio de notificación de eventos está basado en un sistema publicador/suscriptor que permite a las aplicaciones interesadas suscribirse a tipos de eventos sobre los cuales desean ser notificadas. Al momento de recibir una notificación para una aplicación suscriptora, Eleggua determina la acción a seguir mediante el uso de reglas ECA (Evento –Condición – Acción) [5]. Dentro de las acciones definidas en las reglas, esta la posibilidad de utilizar Web Services como mecanismo de comunicación request/reply entre aplicaciones y permitir la transferencia de información entre las mismas. Las acciones también permiten definir llamados a APIs de las aplicaciones participantes y la generación de nuevos eventos lógicos.

En la figura 6 se ilustran los principales componentes de Eleggua. El middleware de eventos distribuidos (DEM por sus siglas en inglés), los *proxies* de cooperación que realizan una labor de intermediación entre las aplicaciones externas y la plataforma de eventos y finalmente un monitor de eventos que permite rastrear el envío y recepción de los eventos entre las aplicaciones y también determinar las excepciones ocurridas durante el proceso.

A continuación se presenta una descripción más detallada de cada uno de los componentes y de las relaciones entre ellos.

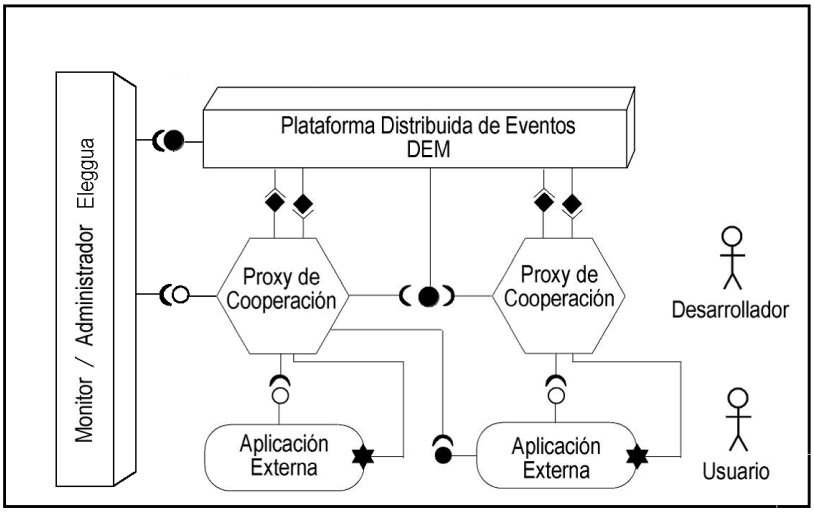


Figura 6: Principales Componentes de Eleggua

4.2.1 La Plataforma de Eventos Distribuidos (DEM)

Esta plataforma proporciona la funcionalidad básica de un sistema publicador/suscriptor para la notificación de eventos. Las aplicaciones presentes en la infraestructura, notifican y reciben eventos (push/pull) a través de sus representantes los Proxies de Cooperación.

DEM está compuesto por tres tipos de componentes distribuidos: componente despachador, componentes consumidores y componentes productores. El componente despachador está implementado como un componente centralizado y su función principal es llevar a cabo la comunicación entre los consumidores y productores. Los componentes consumidores ofrecen un API para la interacción con DEM para realizar el consumo de eventos. Este API ofrece las funciones necesarias para el registro de aplicaciones consumidoras, registro como suscriptor de eventos de un tipo determinado y mecanismos push/pull para el consumo de un evento. Los componentes productores ofrecen, de la misma manera que los consumidores, un conjunto de funciones para la interacción con DEM, tales funciones permiten el registro de una aplicación como productora de eventos, la suscripción a un tipo de eventos y mecanismos para el envío (push) de eventos generados.

Todos los envíos de mensajes desde una aplicación externa, como resultado de operaciones tales como envío de suscripciones, registro como publicador y generación de eventos, son almacenados localmente en el respectivo componente productor o consumidor para su interacción con el despachador. El despachador, una vez recibido el mensaje, hace las validaciones necesarias para continuar con el proceso, por ejemplo, el envío de un evento por parte de

un publicador, exige que con anterioridad la aplicación se haya registrado con el despachador con este rol para un tipo de evento determinado. Igualmente para llevar a cabo la notificación a los suscriptores, estos previamente deben haberse registrado con el despachador para ser notificados.

Un componente consumidor guarda una copia local de todos los eventos para cada aplicación interesada, organizados por tipo de evento para que puedan ser consumidos mediante el API del componente. Finalmente, DEM ofrece servicios para la consulta de eventos históricos y filtros sobre eventos producidos por determinados productores y consumidos por consumidores específicos. Estos servicios son implementados mediante Web Services y pueden ser utilizados por aplicaciones externas como mecanismo de reacción ante la llegada de un evento o como medio de monitoreo y control para el monitor de eventos de la infraestructura.

4.2.2 Proxy de Cooperación

Un Proxy de cooperación o CP tiene las siguientes responsabilidades:

- Observar la aplicación que representa (figura 7a) para notificar al DEM, en forma de eventos lógicos, observaciones sobre las ejecuciones que se realicen sobre la aplicación. En la figura 7b, la relación de notificación de eventos lógicos es identificada mediante rombos sombreados.

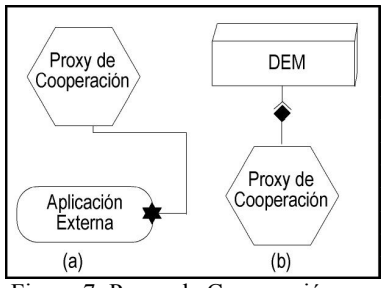


Figura 7: Proxy de Cooperación

- Recibir notificaciones de eventos, producidos por otras aplicaciones, por parte del DEM. En la figura 8, la relación de recepción de eventos lógicos es identificada mediante rombos sombreados.

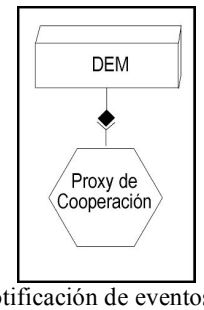


Figura 8: Notificación de eventos al CP

- Procesar las reglas ECA. Este procesamiento está determinado por las acciones a llevarse a cabo, particularmente con las aplicaciones externas, estas actividades pueden ser la utilización de un API mostrado en la figura 9a o la utilización de un Web Service mostrado en la figura 9b.

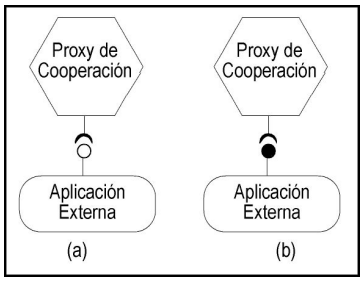


Figura 9: Interacción de un proxy de cooperación con una aplicación externa

Un Proxy de cooperación consta de tres componentes principales: componente observador interno, componente receptor de eventos y un componente procesador de eventos.

El componente observador interno ofrece servicios para la observación y generación de eventos, mediante el registro de reglas de observación, observación de aplicaciones y generación de eventos lógicos. Este componente lleva a cabo la observación de aplicaciones mediante la intercepción de código, utilizando programación por aspectos [3]. El componente receptor de eventos tiene como funcionalidad principal el registro de aplicaciones, suscripción a tipos de eventos y manejo de notificación de eventos lógicos. El componente procesador de eventos ofrece servicios para la declaración y ejecución de reglas ECA, en donde cada regla especifica un tipo lógico de evento, una condición y una acción a ser ejecutada.

4.2.3 Monitor / Administrador Eleggua

Eleggua cuenta con un componente de monitoreo y administración que permite llevar a cabo labores de control sobre la plataforma de eventos, así como la generación de alarmas y manejo de excepciones en caso de presentarse alguna falla en la ejecución de una acción. Mediante el monitor de eventos los administradores de la plataforma pueden consultar que aplicaciones han generado eventos y que aplicaciones los han consumido, así como posibles errores y excepciones que se hayan generado o simplemente hacer consultas históricas sobre los tipos de eventos generados en un periodo determinado. El componente ofrece servicios para el control y administración de las aplicaciones externas, los tipos de eventos y las subscripciones, los eventos, las reglas ECA y de observación, y el desempeño y configuración de los componentes.

4.3 Implementación de las Herramientas de Apoyo

La infraestructura presentada está implementada sobre la plataforma J2EE, específicamente sobre el contenedor de Jboss[23]. Los componentes que conforman los proxies de cooperación y la plataforma distribuida DEM, fueron implementados mediante el uso EJBs[10] de tipo Entity, Session y Message Bean. El manejo de eventos distribuidos se hizo utilizando la especificación JMS[22], y mediante el uso de conexiones punto a punto entre los componentes para el registro y suscripción de aplicaciones. Para la entrega de un evento a un suscriptor, se establece una comunicación entre el componente despachador y el consumidor, todos los mensajes intercambiados son guardados dentro de un mensaje JMS mediante la implementación de JbossMQ[24]. Para la implementación del monitor se utilizan Managed Beans los cuales interactúan directamente con los EJBs de sesión y de entidad de los otros componentes.

Para llevar a cabo la observación de las aplicaciones externas y poder capturar la ejecución de métodos específicos, se utiliza AspectJ[2]. Mediante el uso de aspectos, es posible agregar la funcionalidad necesaria para la comunicación de un evento a la plataforma DEM sin necesidad de cambiar el código de la aplicación externa antes de compilación.

5 RESULTADOS OBTENIDOS

El proceso propuesto está siendo probado, en un ambiente controlado de producción, por la empresa de desarrollo de software Heinsohn Software House S.A., los resultados arrojados por esta prueba muestran que efectivamente es posible llevar a cabo la coordinación del proceso de pruebas en un equipo de trabajo distribuido ya que al definir explícitamente el proceso de pruebas y asociar este proceso con reglas de integración de aplicaciones, disminuye la necesidad de una comunicación entre las personas para tratar de sincronizar las actividades. Igualmente se ha podido comprobar que la plataforma distribuida de eventos ha permitido la interacción de las aplicaciones desarrolladas y de aplicaciones de administración interna del cliente sin necesidad de cambiar nada en el código de las mismas.

Finalmente la herramienta de planeación y seguimiento permite que los involucrados sepan exactamente cuáles son sus responsabilidades y las tareas que deben realizar. Muchas de estas tareas son creadas automáticamente gracias a la infraestructura de integración lo que facilita la sincronización de trabajo.

De los resultados obtenidos hasta el momento surge la necesidad de trabajar en un lenguaje de más alto nivel, cercano al proceso, para la definición de las reglas de integración. También, hemos podido validar la infraestructura como tal e identificar requerimientos de administración y monitoreo más específicos de los que hasta el momento habíamos realizado.

6 TRABAJOS RELACIONADOS

Para resolver el problema de comunicación y coordinación de aplicaciones en ambientes distribuidos, se han planteado varias alternativas. Las más usuales son aquellas relacionadas con el manejo de comunicaciones asincrónicas, asistidas

por plataformas de manejo de eventos, en las que se sigue el modelo publicador/suscriptor para la notificación de los mismos. Algunos de los trabajos relacionados más relevantes utilizados para nuestra investigación fueron JEDI[8], SIENA[32], EVE [13], EDEM [17] y Hermes[30]

JEDI, desarrollado en el politécnico de Milano es un sistema de notificación por eventos usado para implementar un sistema de administración de workflow distribuido; el enfoque principal del grupo es la distribución y migración de las aplicaciones.

SIENA, desarrollado por la Universidad de Colorado, es un sistema de notificación por eventos de escala de Internet; su consideración principal es el impacto del diseño sobre el funcionamiento en redes WAN.

EVE, desarrollado en la Universidad de Zurich, es un sistema de workflow extensible que usa una infraestructura basada en eventos. El sistema propuesto provee un framework basado en componentes, esta aproximación les permite componer y reutilizar las entidades del proceso en un sistema coherente. El sistema es construido extendiendo la funcionalidad de una plataforma de integración basada en eventos. Esta plataforma provee funcionalidad de registro, detección y administración de eventos, notificación a componentes distribuidos y autónomos, y administración y monitoreo del proceso a través de historiales de eventos. La plataforma se basa además en el uso de componentes reactivos cuya interacción es caracterizada por reglas evento-condición-acción (ECA).

EDEM, desarrollado por la Universidad de California en Irving, es una infraestructura de aplicación y usabilidad basada en eventos que se enfoca en la observación de aplicaciones a través de *probes* y la generación de eventos basada en esta observación.

HERMES es una plataforma basada en eventos desarrollada en la Universidad de Cambridge, especialmente diseñada para dar soporte al desarrollo de aplicaciones distribuidas a gran escala. Está implementada sobre una capa de enrutamiento punto a punto que permite llevar a cabo el enrutamiento de eventos basados en su contenido, que facilitan la diseminación de eventos desde los publicadores a los suscriptores.

7 CONCLUSIONES Y TRABAJOS FUTUROS

La propuesta presentada en este artículo responde a algunas de las necesidades identificadas para dar soporte adecuado a los grupos de desarrollo de software geográficamente dispersos. Específicamente en este proyecto nos hemos concentrado en el proceso, las herramientas y la infraestructura de integración que permita coordinar procesos de pruebas y corrección de defectos en ambientes GSD. Adicionalmente, hemos tenido en cuenta la restricción adicional de la interoperabilidad de las diferentes aplicaciones utilizadas, que debe lograrse de una manera desacoplada ya que las aplicaciones son independientes.

Los objetivos de proveer a grupos de desarrollo dispersos, con un proceso que facilite su operación y herramientas de soporte que permitan la ejecución del proceso se cumplieron. Se desarrollaron herramientas que permiten implementar un modelo de desarrollo en V en un equipo de trabajo geográficamente disperso. Estas herramientas facilitan la creación de requerimientos, planes de pruebas basados en dichos requerimientos y la coordinación de las tareas asociadas a cada una de estas actividades. Igualmente, se desarrolló una plataforma distribuida de eventos que permite llevar a cabo la observación y notificación de eventos como mecanismo de intercambio de información entre las aplicaciones de soporte al proceso de pruebas.

Como aspecto a mejorar se encontró que la utilización de AspectJ como mecanismo de observación de las aplicaciones impone restricciones tales como tener las clases Java de las aplicaciones disponibles para programar su intercepción.

Finalmente, es importante resaltar que el uso de esta tecnología impone un reto no solo técnico sino organizacional para las empresas, ya que involucra un cambio de cultura de parte de los participantes.

Como continuación a este proyecto se va a trabajar en el componente de monitoreo y administración para agregarle funcionalidad que brinde un mayor apoyo a los administradores de la plataforma y a los desarrolladores de la integración, aprovechando los mecanismos ofrecidos por la tecnología JMX[21]. Adicionalmente se trabajará en la integración de herramientas de planeación y gestión de proyectos tales como MS-Project[28] para que pueda existir una integración las tareas generadas en Cronos.

Referencias

[1] Agile Modeling. http://www.agilemodeling.com/. [Last visited on 2005-01-13]

[2] AspectJ Project. http://eclipse.org/aspectj/. [last visite don 2005-08-05]

[3] Aspect Programming: New Perspectives on software. http://www.aspectprogramming.com/. [last visited on 2005 08-05]

[4] Battin Robert et Al. "Leveraging Resources in Global Software Development". IEEE Software Marzo 2001.

[5] Bae J. et Al. "Automatic control of workflow Processes using ECA rules". IEEE Transactions on knowledge and data engineering 16 (8). Agosto 2004

[6] Capability Maturity Model for Software (SW-CMM). http://www.sei.cmu.edu/cmm/ [last visited on 2005-08-05]

[7] Carmel Erran & Agarwal Ritu. "Tactical Approaches for alleviating distance in global software development". IEEE Software Marzo 2001.

[8] Cugola G, Nitto E. & Fuggeta A. "The JEDI Event Based Infraestructure and Its Application on the Development of the OPSS WFMS". IEEE Transactions on Software Engineering (27) 2001.

[9] Ebert Christof & De Neve Philip. "Surviving Global Software Development". IEEE Software Marzo 2001.

[10] Enterprise Java Beans Technology http://java.sun.com/products/ejb/ [last visited on 2005-08-05]

[11] Extreme Programming: A gentle introduction. http://www.extremeprogramming.org/ [last visited on 2005-08-05]

[12] Fowler Martin. Using an Agile Software Process with Offshore Development http://www.martinfowler.com/articles/agileOffshore.html [last visited on 2005-01-13]

[13] Geppert A. & Tomboros D. "Event-based Distributed Workflow Execution with EVE". In Proc. IFIP Int'l Conf. on Distributed Systems Platforms and Open Distributed Processing (Mideware'98), Lake District, England, Septembre 1998

[14] Herbsleb James & Deependra Moitra . "Global Software Development" IEEE Software, Marzo 2001.

[15] Herbsleb James et Al. "An empirical study of global software development: distance and speed" Procedings of the 23rd International conference on Software Engineering. Toronto 2001.

[16] Herbsleb James & Mockus Audris. "Formulation and Preliminary Test of an empirical Theory of Coordination in Software Engineering".ESEC/FSE'03 Septiembre 2003

[17] Hilbert D. & Redmiles D. "An Approach to Large-Scale Collection of Application Usage Data Over the Internet". In Proc. 20th International Conference on Software Engineering. 1998

[18] Heeks R. et Al. "Synching or Sinking: Global Software Outsorcing Relationships". IEEE Software. Marzo 2001.

[19] Humphrey W. " Introduction to the Team Software Process". Addison-Wesley. 1999

[20] IBM – Racional Unified Process - http://www.agilemodeling.com/ [last visited on 2005-01-13]

[21] Java Management Extensions (JMX). http://java.sun.com/products/JavaManagement/ [last visited on 2005-08-05]

[22] Java Message Service. http://java.sun.com/products/jms/ [last visited on 2005-08-05]

[23] JBOSS Application Server – http://www.jboss.org [last visited on 2005-01-15

[24] JbossMQ http://www.jboss.org/wiki/Wiki.jsp?page=JBossMQ [last visited on 2005-01-15]

[25] Kaner, Cem "Quality Cost Analysis: Benefits and Risks", http://www.kaner.com/qualcost.htm [Last visited on 2003-05-01]

[26] Mann, Charles, "Why Software is so Bad", Technology Review, http://www.technologyreview.com [last visited on 2003-05-4]

[27] Mockus Audris & Weiss David. "Globalization by chunking: A quantitative approach". IEEE Software Marzo 2001

[28] Microsoft Project. http://office.microsoft.com/en-us/FX010857951033.aspx [last visited on 2005-08-05]

[29] Mohagheghi P. "Global Software Development: Issues, Solutions, Challenges". University of Science and Technology (NTNU), Norway.

[30] Pietzuch, Peter. " Event-Based Middleware: A New Paradigm for Wide-Area Distributed Systems". Cambridge

Unvierstiy. 2002

[31] Sosa M. et Al. "Factors that influence technical communication in distributed product development: An empirical study in the telecomunications industry". IEEE Transactions on Engineering Management 49(1). Febrero 2002.

[32] Tarkoma S. "Scalable Internet Event Notification Architecture (SIENA)" Research Seminar on middleware for mobile computing, University of Helsinki 2002.

[33] V-MODEL. http://www.kbst.bund.de/V-Modell [last visited on 2005-12-01]

Modelo para estimación de la calidad de un Web Service

María Pérez, Luis E. Mendoza, Anna C. Grimán,
Universidad Simón Bolívar, Departamento de Procesos y Sistemas, LISI
Caracas – Venezuela, Apartado Postal 89000, Z.P. 1080-A
{movalles, lmendoza, agriman}@usb.ve

Resumen

Los Web Services ofrecen una alternativa de *software* independiente en cuanto a la plataforma, basada en estándares para la integración de aplicaciones, la automatización de procesos de negocio y la publicación de la información de diversas fuentes. La calidad en Web Service es vital para una organización que busca apalancarse en la integración. Contar con una infraestructura integrada, segura, escalable y disponible disminuye costos y permite compartir información de manera confiable. Una forma de responder esta necesidad es valorando el nivel de calidad del Web Service, por lo cual se hace necesario desarrollar un *Modelo de Estimación de la Calidad* con el uso de criterios de evaluación considerados como estándares, aplicados al caso específico de Web Service. Este modelo responde a la necesidad de disponer de una herramienta que estime la calidad de estos productos en una manera específica. Las empresas podrán contar así con un mecanismo confiable que permita determinar los aspectos de calidad relevantes de los productos desarrollados. Este modelo fue evaluado a través de un Análisis de Características en dos estudios de caso, arrojando que el mismo es pertinente, completo, adecuado y preciso.

Palabras Claves: Modelo de Calidad, Web Service

Abstract

Web Services offer an alternative of independent software as far as the platform, based on standards for the integration of applications, the business processes automatization and the publication of the information of diverse sources. A high-quality Web Service is medullar for an organization demanding integration support. Counting with an integrated, safe, scaleable and available infrastructure diminishes costs and allows sharing information in a reliable way. A form to respond this necessity is valuing the quality level of the Web Service. In order to accomplish this, a Model *of Quality Estimation is developed by using* standard evaluation criteria, applied to the specific case of Web Service. This model responds to the necessity to have a tool that considers the quality of these products in a specific way. Thus, companies will be able to count on a reliable mechanism that allows determining the excellent aspects of quality of developed products. This model was evaluated through Feature Analysis in two case studies, concluding that it is pertinent, complete, suitable and precise.

Keywords: Quality Model, Web Service

1. INTRODUCCIÓN

La estimación de la calidad de software es, en general, un proceso difícil de llevar a cabo ya que requiere de un compromiso por parte de los agentes externos a la organización a evaluar y por parte de sus miembros, los cuales orientan la evaluación según sus necesidades. En particular, este tópico empleado en tecnologías novedosas como son los Web Service (WS) se convierten en un reto. En este sentido, la presente investigación propone una adecuación del *Modelo Sistémico de Calidad* (MOSCA) [18], para que éste sea aplicable a la nueva tecnología que impulsa a las empresas: los WS. Los WS ofrecen una alternativa de software independiente en cuanto a la plataforma, basada en estándares para la integración de aplicaciones, la automatización de procesos de negocio y la publicación de información de diversas fuentes e intercambio empresarial. Lo que lleva a preguntarse es si en realidad se están desarrollando de manera provechosa para las organizaciones. Poseer un WS de calidad es de importancia para una organización ya que se benefician las relaciones de negocio contribuyendo tanto a la integración interna como la inter-empresarial; al asegurar una infraestructura integrada, segura, escalable y disponible se disminuye el costo de propiedad y permite

compartir información de manera confiable.
Para ello se propone un Modelo de Estimación de la Calidad de Web Service basado en MOSCA y en el estándar Calidad de Servicio (Quality of Service, QoS), como una solución al problema de no disponer de una herramienta que valore la Calidad de los WS. El principal aporte de este modelo es que parte de una base ya probada (MOSCA), el cual a su vez se inspira en estándares de calidad y propone métricas, para luego incorporar los aspectos propios de la calidad de Web Service y QoS, donde éste último es también una fuente de requerimientos importante íntimamente relacionados al mundo de la Ingeniería Web y las aplicaciones en Internet. En consecuencia, las empresas que aplican esta tecnología podrán conocer los atributos de calidad que están presentes en su sistema (confiabilidad y seguridad, entre otros) y las recomendaciones oportunas que favorecen su desarrollo. Como logro adicional, se evaluó el modelo propuesto con dos estudios de caso, encontrándose que el mismo es pertinente, completo, adecuado y preciso.
A parte de la Introducción y las Conclusiones, este artículo está conformado por cuatro secciones adicionales, la primera describe los antecedentes de la investigación, en la segunda se identifican las características de calidad para los WS, en la tercera se propone el modelo de calidad y en la cuarta se describe la evaluación del modelo con los resultados más relevantes.

2. MARCO REFERENCIAL

La calidad según ISO/IEC 9126 [9] es la totalidad de las características de una entidad que refieren su capacidad de satisfacer necesidades establecidas o implícitas. La revisión bibliográfica [4, 5, 8, 9, 10, 11, 26] refleja la tendencia de formular modelos que especifiquen la calidad del software. Tomando en cuenta la calidad del producto y la calidad del proceso, se desarrolló MOSCA [18], que integra el modelo de calidad del producto [20] y el modelo de calidad del proceso de desarrollo [22], y está soportado por los conceptos de calidad total sistémica [2, 21]. En cuanto a la calidad del Producto, MOSCA plantea, sobre la base de las 6 características de calidad del estándar internacional ISO/IEC 9126 [9], un conjunto de categorías, características y métricas asociadas, que especifican la calidad de un producto de software con un enfoque sistémico y hacen del modelo un instrumento de medición de valor ya que cubre todos los aspectos imprescindibles para medir directamente la calidad del producto de software.
La literatura [6, 13, 16, 19, 25] precisa que los WS son aplicaciones multiplataforma basadas en componentes interoperables que utilizan estándares de protocolos de Internet; además, permiten crear una única puerta de entrada a la organización exponiendo las funcionalidades del negocio sobre la Web, facilitando así la integración del mismo.
La Figura 1 muestra la arquitectura conceptual orientada a servicio (Service Oriented Architecture, SOA) usando las tecnologías: Protocolo de Acceso de Objeto Simple (Simple Object Access Protocol, SOAP), Lenguaje de Descripción de los Servicios Web (Web Services Description Language, WSDL) e Integración, Descubrimiento y Descripción Universal (Universal Description, Discovery and Integration, UDDI) y mostrando los servicios que la conforman en el marco de los WS.

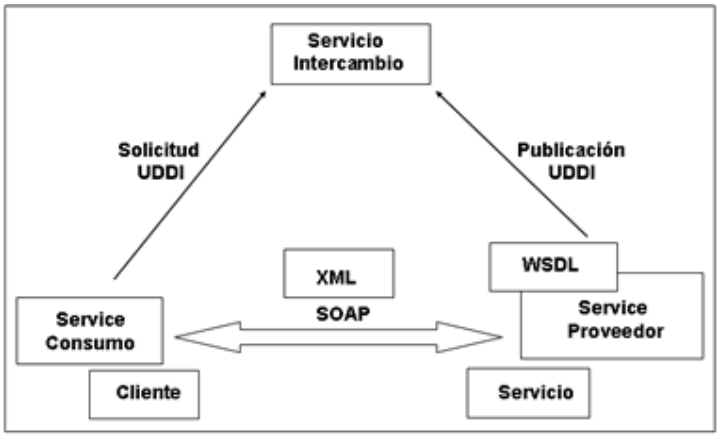


Figura 1. Vista de implementación de SOA con las tecnologías SOAP, WSDL, y UDDI [25].

Según IEEE, QoS para aplicaciones que deben comunicarse en tiempo real, es el conjunto de características cualitativas y cuantitativas de un sistema multimedia distribuido, que son necesarias para lograr la

funcionalidad requerida en una aplicación [7]. En este sentido, QoS determina la usabilidad y utilidad del servicio, influenciando en la popularidad del mismo. Con la extensa propagación de los WS, QoS se ha convertido en un factor significante para determinar el éxito que pueda alcanzar un proveedor de servicio sobre otros [14]. Aplicar QoS en Internet es un reto crítico y significante debido a la naturaleza dinámica e impredecible del Web.. QoS cubre un rango de técnicas que armoniza las necesidades de quien requiere el servicio con quien lo provee, basándose en los recursos de red disponibles [14]. A través de QoS se puede hacer referencia a las propiedades no funcionales del WS, tales como rendimiento, fiabilidad, disponibilidad y seguridad [14], las cuales representan los requerimientos de calidad necesarios para alcanzar la funcionalidad esperada del mismo.

3. WEB SERVICES Y CARACTERÍSTICAS DE CALIDAD

Los WS pueden tener varias formas de clasificarse. Los autores conciben diferentes tipos de WS que no se contradicen entre sí. Según la funcionalidad de los Web services: Canta y sus colegas [3] dividen a los WS en dos categorías: WS Programáticos (Programmatic Web Services, PWS) y WS Interactivos (Interactive Web Services, IWS); los primeros encapsulan la funcionalidad en la capa de negocios de las aplicaciones, mientras que los segundos la encapsulan en la interfaz de usuario o la capa de presentación. Los PWS extienden la capa lógica de negocios de una aplicación. [3]. En los WS la característica de mantenibilidad se ve reflejada a través de la simpleza de las operaciones, que permiten la facilidad de cambio, análisis y pruebas; además propicia funcionalidad gracias a la posibilidad que posee el suscriptor de adecuar dichas funciones a sus requerimientos específicos. Los IWS exponen una interfaz de usuario de aplicación. Como los IWS se manejan a través de la interfaz del usuario, la característica de usabilidad se ve reflejada en ellos.

Según las necesidades del negocio. La empresa Microsoft clasifica a los WS según la funcionalidad que el negocio desee darle al mismo:

3.1 Los WS Orientados a datos (Web Services Data Centric, WSDC) que son útiles en situaciones donde deben actualizarse datos que son alterados frecuentemente. [15]Se puede deducir que los WSDC propician la característica de eficiencia al negocio; por lo que las organizaciones que requieren de la adecuada utilización del tiempo y los recursos, ven conveniente la aplicación de este tipo de WS para obtener los beneficios esperados.

3.2 Los WS de Colaboración (Web Service Collaboration, WSC) son aquellos que permiten la colaboración entre todos los involucrados del negocio [15]. Estos WS suministran la característica de interoperabilidad al negocio.

3.3. Los WS para Análisis (Web Service Analysis, WSA) son aquellos que reciben informes de varias empresas filiales, los ejecuta, consolida y entrega los resultados deseados [15]. Por lo tanto, este tipo de WS impulsa la característica de usabilidad del negocio.

3.4 Los WS de Alertas (Web Services Alert, WSAL) manejan situaciones de alerta más eficientemente, ya que se puede enviar mensajes de aviso a un dispositivo móvil (celular, pda, etc) para que se pueda actuar de inmediato [15].La característica fundamental que propicia al negocio este tipo de WS, es la eficiencia.

Los requerimientos de los usuarios establecen los criterios para seleccionar el tipo de WS que se ha de aplicar. Estos requerimientos exigen características de calidad que el usuario espera del WS. La tabla 1 resume la clasificación planteada anteriormente junto con la(s) característica(s) de calidad asociada(s) a cada una de ellas y traducidas al estándar ISO 9126 según [12].

Tipo de WS	Característica de calidad del negocio	Según ISO/EIC 9126
PWS	Mantenibilidad, Funcionalidad	Mantenibilidad, Funcionalidad
IWS	Usabilidad	Usabilidad
WSDC	Eficiencia	Eficiencia
WSC	Interoperabilidad	Funcionalidad /Interoperabilidad
WSA	Usabilidad	Usabilidad
WSAl	Eficiencia	Eficiencia
QoS	Rendimiento, Fiabilidad, Seguridad	Eficiencia. Fiabilidad, Funcionalidad/Seguridad

Tabla 1. Características de Calidad asociadas al tipo de WS.

4 MODELO DE CALIDAD PARA WEB SERVICE

Para el desarrollo del producto de calidad aquí propuesto, se partió de MOSCA [18]. Este modelo fue desarrollado por el Laboratorio de Información y Sistemas de Información (LISI) de la Universidad Simón Bolívar, con el financiamiento del gobierno venezolano. Ha sido probado en más de seis estudios de caso

[23]. Se basa en un enfoque de calidad sistémica el cual contempla tanto la perspectiva producto (software) como la perspectiva proceso. Con base a esta experiencia y a su condición sistémica, se decidió hacer una adaptación del mismo para WS. Para ello se siguió la metodología propuesta por Rincón et al. [24]. Como consecuencia de esta adaptación, se utiliza sólo la perspectiva de producto, de ella se adaptaron tanto las categorías como las características; finalmente se propusieron 46 métricas. Adicionalmente, se compararon las características de calidad identificadas en la Tabla 1, concluyendo que se utilizarían las seis que presenta el MOSCA ya que son las mismas

A continuación se describe el modelo obtenido, el cual se puede apreciar en la Figura 2.

Nivel 0: Dimensiones. Aspectos Internos y Aspectos Contextuales del producto.

Nivel 1: Categorías. En el modelo se contemplan 6 categorías las cuales se describen en la Tabla 2.

Nivel 2: Características. Cada categoría tiene asociado un conjunto de características, las cuales definen las áreas claves que se deben satisfacer para lograr, asegurar y controlar la calidad de un WS. Las Tablas 3 a 8 muestran cada categoría con sus características asociadas. Las categorías de Usabilidad, Eficiencia y Fiabilidad, y sus correspondientes características, fueron adecuadas a WS.

Nivel 3: Métricas. Para cada característica se propone un conjunto de métricas que se formularon según el paradigma Meta-Pregunta-métrica (Goal Question Metric, GQM) de Basili [1], el cual fue útil al proporcionar un enfoque sistemático que permite ir desde unos requerimientos u objetivos, hasta las mediciones necesarias para valorar directamente los mismo. En este sentido, las metas pueden ser vistas como los atributos de calidad que se quieren medir. Por otra parte, para cada característica se orienta sobre la forma en que las respuestas deben ser dadas para poder determinar la evaluación final del WS. Las métricas propuestas se encuentran detalladas en la Tabla 9.

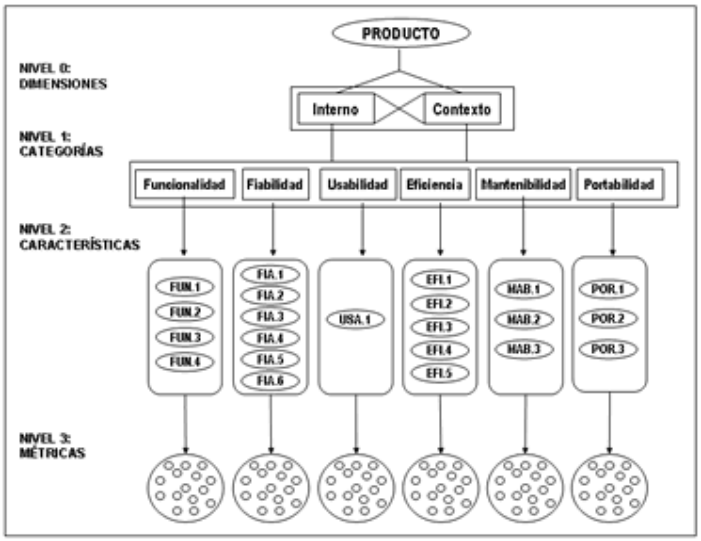


Figura 2. Representación gráfica de la especificación de MOSCA para WS.

Categorías	Definición
Funcionalidad (FUN)	Es la capacidad del Webservice para proveer las funciones que cumplan con las necesidades específicas o implícitas, cuando es utilizado bajo ciertas condiciones.
Fiabilidad (FIA)	La fiabilidad es la capacidad del Webservice para mantener un nivel especificado de rendimiento cuando es utilizado bajo condiciones especificadas.
Usabilidad (USA)	Esta categoría se refiere a la capacidad del Webservice para ser atractivo, entendido, aprendido y utilizado por el usuario bajo condiciones específicas.
Eficiencia (EFI)	Es la capacidad del Webservice para proveer un rendimiento apropiado, relativo a la cantidad de recursos utilizados, bajo condiciones específicas.
Mantenibilidad (MAB)	Es la capacidad del Webservice para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software ante cambios del ambiente, en requerimientos y especificaciones funcionales.
Portabilidad (POR)	La portabilidad es la capacidad del Webservice para ser transferido de un ambiente a otro

Tabla 2. Categorías del Modelo de Calidad de Webservices.

Características	Descripción
FUN.1 Precisión	Capacidad del Webservice para proveer resultados o efectos correctos o convenientes. Esto incluye el grado de precisión de los valores calculados. Está relacionada con la correctitud en la ejecución de las transacciones.
FUN.2 Transacción	Capacidad de tratar una secuencia de actividades como una simple unidad de trabajo.
FUN.3 Seguridad	Capacidad del Webservice de proteger su información y datos así como la de controlar el acceso no autorizado al mismo
FUN.4 Interoperabilidad	Capacidad del Webservice para interactuar con uno o más sistemas (Propiciada por la tecnología – XML).

Tabla 3. Características del Modelo correspondiente a la categoría Funcionalidad.

Características	Descripción
EFI.1 Estandarización	Capacidad del Webservice para ajustarse a estándares, convenciones o regulaciones.
EFI.2 Comportamiento del tiempo	Capacidad del Webservice para proveer respuestas y tiempos de procesamiento apropiados en tiempo de ejecución bajo condiciones específicas.
EFI.3 Utilización de los recursos	Capacidad del Webservice para utilizar cantidades apropiadas de los recursos cuando el mismo ejecuta sus funciones bajo condiciones específicas.
EFI.4 Latencia	Tiempo transcurrido entre enviar una solicitud y recibir una respuesta.
EFI.5 Throughput	Capacidad del Webservice para representar el número de peticiones atendidas por un WS en un período de tiempo determinado.

Tabla 4. Características del Modelo correspondiente a la categoría Eficiencia.

Características	Descripción
FIA.1 Disponibilidad	Capacidad del producto del software de estar siempre presente y en estado operativo al momento de ejecutar una función en un período determinado, bajo condiciones específicas. Está relacionada con el tiempo de reparo.
FIA.2 Tiempo de Reparo	Representa el tiempo que toma reparar un servicio que ha fallado.
FIA.3 Accesibilidad	Según QoS, representa la capacidad de un servicio para atender una solicitud de un WS. Puede ocurrir que un WS este disponible más no accesible. Está relacionada con sistemas altamente escalables
FIA.4 Escalabilidad	Habilidad de atender consistentemente las solicitudes a pesar de las variaciones del volumen de la demanda.
FIA.5 Tolerancia a Fallas	Capacidad del producto de software para mantener un nivel de rendimiento específico en caso de errores en el software o de infracciones sobre sus interfaces.
FIA.6 Madurez	Capacidad del producto de software para evitar fallas como resultado de errores en el mismo.

Tabla 5. Características del Modelo correspondiente a la categoría Fiabilidad.

Características	Descripción
USA.1 Documentación	Este concepto está relacionado con la existencia de un documento WSDL donde se expone la funcionalidad, la accesibilidad y la comunicación del WS, entre otros.

Tabla 6. Características del Modelo correspondiente a la categoría Usabilidad.

Características	Descripción
MAB.1 Capacidad de análisis	Capacidad del producto de software para ser diagnosticado por deficiencias o fallas en el software.
MAB.2 Capacidad de cambio	Capacidad del producto de software para implementar una modificación específica de una manera más sencilla.
MAB.3 Estabilidad	Capacidad del producto de software para evitar efectos inesperados después de modificaciones en el software.

Tabla 7. Características del Modelo correspondiente a la categoría Mantenibilidad.

Características	Descripción
POR.1 Adaptabilidad	Capacidad del producto de software para ser adaptado a diferentes ambientes especificados sin aplicar acciones u otros medios que no sean los provistos para este propósito en el software considerado.
POR.2 Co-existencia	Capacidad del producto de software para co-existir con otro software independiente en un ambiente común compartiendo recursos comunes.
POR.3 Capacidad de reemplazo	Capacidad del producto de software para ser usado en lugar de otro producto de software especificado para el mismo propósito en un mismo ambiente. Por ejemplo, la capacidad para el reemplazo de una nueva versión de un producto es importante para el usuario, cuando ésta se actualiza.

Tabla 8. Características del Modelo correspondiente a la categoría Portabilidad.

Característica	Métrica	Pregunta	Tipo de escala
1.1. Precisión	Resultados o efectos correctos o convenientes	Tasa de respuestas correctas.	1
	Ejecución de las actividades	Tasa de actividades que se ejecutan completamente.	2
1.2. Transacción	Correspondencia del estado inicial	¿Cuándo ocurre un error durante una transacción, el estado inicial se mantiene consistentemente?.	4
	Correspondencia del estado final	¿Cuándo una transacción es exitosa, el estado final perdura en el tiempo; es decir; es persistente?.	4
	Cantidad de transacciones exitosas	Tasa de transacciones son completadas exitosamente.	5
1.3.Seguridad	Seguridad de la comunicación	¿El WS utiliza el protocolo SSL (Secure Sockets Layer) para transmitir los datos?.	3
		¿El WS posee algún certificado criptográfico SSL proporcionado por una Autoridad Certificada?.	3
		¿Existe algún mecanismo que permite proteger el certificado SSL?.	3
		¿El WS genera alguna copia de la información confidencial que se transmite?.	3
	Seguridad de la aplicación	¿El WS utiliza Apache como servidor Web?.	3
		¿El WS fue desarrollado utilizando tecnología JAVA?.	3
		¿El WS se encuentra alojado en un lugar físicamente seguro?.	3
	Protección del código fuente del WS	¿El WS posee algún mecanismo que permite proteger su propio código fuente de usuarios no autorizados?.	3
	Control del acceso no autorizado	¿El WS posee algún mecanismo para controlar el acceso no autorizado?.	3
	Autentificación	¿El WS posee algún mecanismo de autentificación de usuarios?.	3
1.4. Interoperabilidad	Existencia de funcionalidades que pertenecen a otro sistema	¿Existen funcionalidades utilizadas por el WS, que pertenecen a otro sistema? .	3
	Existencia de funcionalidades pertenecientes al WS que son utilizadas por otros sistemas	¿Existen funcionalidades utilizadas por otros sistemas, que pertenecen al WS? .	3
	Tasa de funcionalidades utilizadas por el WS, que pertenecen a otro sistema	Tasa de funcionalidades que pertenecen a otros sistemas.	6
	Intercambio de datos	¿Existe intercambio de datos con otros sistemas?.	3
2.1. Estandarización	Ajuste a estándares, convenciones o regulaciones.	¿EL WS se ajusta a los estándares XML, SOAP, UDDI y WSDL?.	3
		¿El WS se ajusta a convenciones o regulaciones existentes?.	3

2.2. Comportamiento del tiempo (Tiempo de respuesta)	Suministro de respuestas y tiempos de procesamiento apropiados en tiempo de ejecución	Tasa del tiempo de respuesta obtenido (real) comparado con el tiempo esperado (ideal).	7
2.3. Utilización de recursos	Velocidad de transmisión de datos	¿Cuál es la velocidad de transmisión del WS?.	8
2.4. Latencia	Medir el tiempo transcurrido entre enviar una solicitud y recibir una respuesta	¿Cuánto tiempo promedio transcurre entre enviar una solicitud y recibir respuesta en un tiempo determinado?.	9
2.5. Troughput	Medir el número de peticiones atendidas por un WS en un período de tiempo determinado	Tasa de Throughput.	10
3.1. Disponibilidad	Capacidad del WS de ser encontrado fácilmente por un sistema	¿El WS puede ser encontrado fácilmente por un sistema?.	3
3.2. Tiempo de reparo (TTR)	Medir el tiempo que toma reparar un servicio que ha fallado	¿Cuánto tiempo toma reparar un servicio que ha fallado?.	11
3.3. Accesibilidad	Capacidad del WS para atender solicitudes	¿El WS es capaz de atender todas las solicitudes que se le pide?.	3
3.4. Escalabilidad	Atención consistente de solicitudes a pesar de las variaciones de la demanda	Tasa de capacidad para atender solicitudes.	12
		Tasa de crecimiento anual.	13
3.5. Tolerancia a fallas	Funcionamiento del WS en caso de errores en el software	¿El WS posee algún mecanismo que permita su funcionamiento aunque existan errores en el software?.	3
3.6. Madurez	Número de fallas como resultado de errores en el WS	¿Cuántas fallas ocurren en el WS en una semana?.	14
	Tiempo promedio entre fallas	Tiempo promedio entre fallas.	14
	Capacidad para resolver fallas	Tasa de fallas ocurridas que son resueltas.	16
4.1. Documentación	Existencia del documento WSDL	¿Existe un documento WSDL donde se describe la funcionalidad del WS?.	3
		¿Existe un documento WSDL donde se describe la accesibilidad del WS?.	3
		¿Existe un documento WSDL donde se describe el mecanismo de comunicación del WS?.	3
5.1. Capacidad de análisis	Facilidad para ser diagnosticado	Grado de facilidad al diagnosticar el software.	17
	Auto explicación del código	¿El código es autoexplicativo?.	3
	Indentación del código	¿El código del WS está indentado correctamente?.	3
	Documentación del código	¿El código del WS está documentado correctamente?.	3
5.2. Capacidad de cambio	Parametrización	¿Es utilizado el pase de parámetros?.	3
	Independencia funcional de los módulos	¿El WS está distribuido en módulos diferentes?.	4
	Diferenciación de las capas: presentación, lógica y datos	¿La capa lógica, presentación y datos son diferenciables entre sí?.	4
	Acoplamiento	¿Un módulo accede al módulo subordinado por él, por medio de una lista convencional de argumentos?.	3
		¿Un módulo accede al módulo subordinado por él, pasando una porción de la estructura de datos?.	3
		¿Un módulo accede al módulo subordinado por él, pasando una variable que controla las decisiones en el segundo?.	3

		¿Los módulos están atados a un entorno externo al software?.	3
		¿Varios módulos hacen referencia a un área global de datos?.	3
		¿Un módulo hace uso de datos o de información de control mantenidos dentro de los límites de otro módulo?.	3
	Cohesión	¿Cada módulo del WS realiza un conjunto de tareas poco relacionadas las unas con las otras?.	3
		¿Cada módulo del WS realiza tareas relacionadas lógicamente?.	3
		¿Cada módulo del WS contiene tareas relacionadas por el hecho de que todas deben hacerse en el mismo intervalo de tiempo?.	3
		¿Cada módulo del WS presenta los elementos de procesamiento relacionados entre sí y deben ejecutarse en un orden específico?.	3
		¿Cada módulo del WS presenta a todos los elementos de procesamiento concentrados en un área de la estructura de datos?.	3
5.3. Estabilidad	Cantidad de variables modificadas con respecto al total de variables en un módulo	Cantidad de variables modificadas con respecto al total de variables en un módulo.	
	Cantidad de variables globales con relación a los módulos que las utilizan	Cantidad de variables globales con relación a los módulos que las utilizan.	18
6.1. Adaptabilidad	Descripción de datos independiente de la plataforma	¿Los datos pueden ser descritos independientemente de la plataforma?.	3
	Plataformas de software donde puede ser operado	Cantidad de sistemas operativos (SO) en los que el WS puede funcionar correctamente.	
6.2. Coexistencia	Co-existencia con otros sistemas	¿Existen otros sistemas compartiendo los mismos recursos?.	3
	Tasa de satisfacción con los productos que co-existen	Tasa de satisfacción respecto a los productos que co-existen en el ambiente.	20
	Integración con otros productos	¿El WS es capaz de interactuar con otros sistemas independientemente de la plataforma?.	3
6.3. Capacidad de reemplazo	Capacidad para remplazar a otro sistema con el mismo propósito	¿Puede el WS remplazar a otro sistema?.	21
	Capacidad para utilizar los mismos datos al remplazar a otro sistema	¿Pueden ser utilizados los mismos datos de un sistema al sustituirlo?.	21

Tabla 9. Métricas para valorar la calidad de los WS.

La Tabla 10 muestra los rangos utilizados en las diferentes escalas.

Escala	Rango
1	Respuestas correctas / Respuestas procesadas
2	Actividades ejecutadas completamente / Actividades totales
3	5 = Si 1 = No
4	5= Siempre, 4= Casi siempre, 3= En ocasiones, 2= Muy poco, 1= Nunca
5	Cantidad de transacciones exitosas / Cantidad de transacciones totales.
6	Funcionalidades que pertenecen a otros sistemas / Funcionalidades del WS
7	Tiempo de respuesta real / Tiempo de respuesta requerido
8	Cant. de datos / 60 seg
9	5 = menos de 15 seg, 4 = entre 16 seg y 30 seg, 3 = entre 31 seg y 1 min, 2 = entre 1 min y 5 min, 1 = más de 5 min
10	Cantidad de peticiones atendidas / número de peticiones
11	5 = menos de 1 min, 4 = entre 1 min y 10 min, 3 = entre 10 min y 20 min, 2 = entre 20 min y 1 hora, 1 = más de 1 hora
12	Cantidad de solicitudes que atiende actualmente / cantidad de solicitudes que es capaz de atender
13	Cantidad de solicitudes que es capaz de atender actualmente / cantidad de solicitudes que es capaz de atender en un año

14	4=Ninguna, 3=Muy pocas, 2=Algunas, 1=Muchas
15	5=Semestral o más, 4=Trimestral, 3= Mensual, 2= Semanal, 1= Diario
16	Cantidad de fallas resueltas / Cantidad de fallas ocurridas
17	5=Muy fácil, 4=Fácil, 3=Promedio, 2=Difícil, 1= Muy difícil
18	5 = Muy baja, 4= Baja, 3 = Media, 2 = Alta, 1 = Muy alta
19	5 = 5 SO, 4 = 4 SO, 3 = 3 SO, 2 = 2 SO, 1 = 1 SO
20	N° de productos requeridos con los que co-existe / N° de productos requeridos
21	5 =Totalmente, 4 = Casi todo, 3 = Medianamente, 2 = Poco, 1 =Ninguno

Tabla 10. Tipos de escala para las métricas.

5. EVALUACIÓN DEL MODELO

Para llevar a cabo la evaluación del Modelo propuesto, se aplicó el Método Análisis de Características [11] en dos Estudios de Caso; los pasos seguidos [11] fueron los siguientes: a) identificación de los criterios de la escala de evaluación, b) aplicación del Modelo de Especificación de calidad para WS a dos proyectos, c) evaluación de los criterios del Modelo de Especificación de calidad para WS, y d) análisis de la valoración y presentación de resultados.

5.1 Identificación de los criterios de la escala de evaluación

Se tomó como referencia los pasos que señalan Kitchenham y Jones [11]; por ello fue necesario establecer un conjunto de aspectos a ser verificados que van desde lo más general, el modelo propuesto, a lo más específico, que son las métricas. Según Kitchenham y Jones existen dos tipos de criterios [11]:

- **Simples.** Se utilizan cuando el criterio se encuentra presente o ausente en el contexto de evaluación.
- **Compuestos.** Son utilizados cuando la existencia o conformidad de un criterio puede ser medida con una escala ordinal.

Para efectos de esta evaluación se utilizaron sólo los criterios o aspectos simples para obligar al evaluador a establecer un criterio claro de aceptación (ver Tabla 11).

Escala
1: significa que el Modelo o métrica tiene el criterio establecido.
0: significa que el Modelo o métrica no tiene el criterio establecido.

Tabla 11. Escala utilizada para evaluar los criterios generales y específicos.

Los criterios más generales evalúan el Modelo en un nivel macro, su definición conceptual se presenta en la Tabla 12.

Criterios General	**Descripción**
Pertinencia del Modelo de Calidad	Se refiere a si el Modelo de Calidad propuesto es pertinente o no dentro del proceso de especificación de la calidad del software.
Completitud de las categorías involucradas	Se refiere a si las Categorías: Funcionalidad, Eficiencia, Fiabilidad, Usabilidad, Mantenibilidad, y Portabilidad; dan cobertura total a los aspectos de calidad de los WS.
Adecuación al contexto	Se refiere a si la especificación de la calidad Modelo propuesto es adecuada en el contexto de la evaluación.
Precisión del nivel de calidad especificado por el Modelo propuesto	Se refiere a si la calidad especificada por el Modelo propuesto en el proyecto piloto fue precisa.

Tabla 12. Criterios Generales a evaluar para el Modelo de Especificación de Calidad de WS.

Una vez especificadas los criterios generales fue necesario establecer un conjunto de criterios que permitan evaluar las métricas del Modelo. Los criterios específicos a considerar se presentan en la Tabla 13.
Culminada la identificación de los criterios que permitirán la evaluación, se ha establecido el criterio de aceptación a utilizar. Se tomó como nivel de aceptación de los criterios un valor del 75%. Este porcentaje de aceptación fue tomado por consenso de los equipos evaluadores y los autores, considerando que es una práctica común en la mayoría de los modelos de calidad.

Criterios Específicos	Descripción
Pertinencia de la métrica	Se refiere a si una métrica es adecuada para medir la existencia o no de la característica donde se encuentra.
Factibilidad de la métrica	Se refiere a si es factible medir la característica propuesta en la métrica dentro del contexto de evaluación.
Nivel de profundidad	Se refiere a si la métrica a verificar tiene el nivel de profundidad adecuado para que el resultado sea relevante.
Escala de la métrica	Se refiere a si la escala propuesta es adecuada para medir la métrica.

Tabla 13. Criterios Específicos a evaluar para el Modelo de Especificación de Calidad de WS.

5.2 Aplicación del Modelo de Especificación de calidad para WS a dos proyectos

Luego se aplicó el Modelo a dos estudios de caso, para ello se utilizaron dos WS. Según Kitchenham y Jones [11], para aplicar este método de evaluación, de debe configurar un equipo evaluador. Para este caso se configuraron dos grupos de cuatro personas, dos de ellas investigadores de LISI y dos desarrolladores de cada WS para cada uno de los estudios de caso. La selección del WS a utilizar para la aplicación del Modelo de Especificación de la Calidad propuesto, dependió de la disponibilidad de acceso a los mismos. En este sentido, se logró el acceso a dos WS, el primero es un WS tipo WSA y el segundo WS es del tipo WSC.

El grupo evaluador especificó la calidad de cada WS utilizando el Modelo propuesto. Para ello se aplicó el mismo algoritmo que se propone para MOSCA [17]. Después, haciendo uso de los criterios generales y específicos propuestos anteriormente, se determinó la calidad del Modelo.

5.3 Evaluación de los criterios de Modelo de Especificación de calidad para WS

Las figuras 3 y 4 muestran los resultados de la evaluación de las **Criterios Generales** y de las **Criterios Específicos** del Modelo.

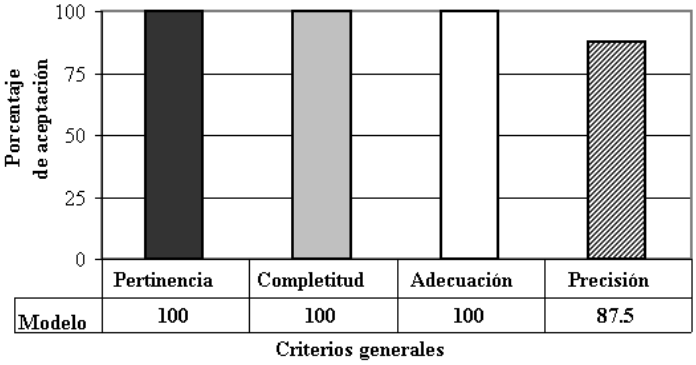


Figura 3 .Resultados criterios generales del modelo.

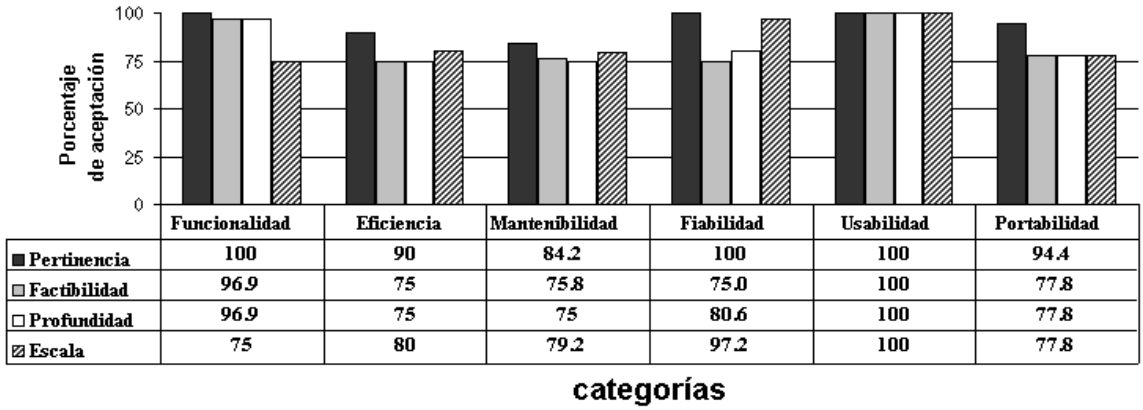


Figura 4. Resultados criterios específicos del modelo por categorías.

5.4 Análisis de la valoración y presentación de resultados

De la aplicación del método de evaluación se obtienen dos tipos de resultados. El primer tipo se refiere al Modelo de Calidad propuesto; es decir, los valores que se obtienen por los criterios. El segundo tipo se refiere a la estimación de calidad que se obtiene sobre los dos WS evaluados.

Sobre la evaluación del modelo se obtiene que todos los criterios generales alcanzaron el máximo porcentaje en la evaluación, a excepción del criterio precisión que alcanzó un 87,5% de satisfacción (Figura 3), considerando que el Modelo de Calidad es **Pertinente** dentro del proceso de especificación de la Calidad del Software. Además, estuvieron de acuerdo en que las categorías Funcionalidad, Eficiencia, Fiabilidad, Usabilidad, Mantenibilidad, y Portabilidad, abarcan **Completamente** los aspectos de calidad de los WS. Siendo parte del contexto de evaluación, los evaluadores consideraron **Adecuada** la calidad del Modelo en dicho contexto.

.En cuanto a la evaluación de cada categoría (Figura 4) se obtuvo que de igual forma cada una por separado es **Pertinente** en cuanto al proceso de estimación de la calidad, es **Completa** en cuanto a su especificación, es **Adecuada** en el contexto de evaluación y **Precisa** en el resultado alcanzado, excepto en la categoría Eficiencia donde la precisión alcanzó un 83% de satisfacción. Esto se debe a que en ciertas métricas de dicha categoría la escala de evaluación es muy subjetiva. Las categorías Funcionalidad, Fiabilidad y Usabilidad fueron consideradas **Pertinentes** en el contexto de evaluación. Le sigue en porcentaje de pertinencia de las métricas la Portabilidad con un 94,43%, luego la Eficiencia resultó obtener un 90% y, por último, la Mantenibilidad con un 84,2% de pertinencia de sus métricas. De acuerdo al criterio de aceptación las categorías resultaron obtener en promedio más del 75% por lo que se les considera aceptables. En el caso de la **Factibilidad** los evaluadores consideraron que en la categoría Usabilidad, el 100% de las métricas eran factibles de evaluar en el contexto en que se aplicaron; seguidamente se encuentra la categoría Funcionalidad con un 96,9% y por último las categorías Portabilidad, Mantenibilidad y Eficiencia y Fiabilidad con un porcentaje de satisfacción mayor o igual al 75%. En el caso de la Profundidad se puede observar en la Figura 5 que para los evaluadores en promedio el 84,21% de las métricas de cada categoría presentaban la **Profundidad** adecuada. Los evaluadores consideraron que el 100% de las métricas de la Usabilidad posee la **Escala** apropiada. Seguidamente las escalas de las métricas para especificar la Funcionalidad, Mantenibilidad, Fiabilidad y Portabilidad, obtuvieron un promedio mayor o igual al 75%. **Puesto que el promedio para cada característica supera ampliamente el 75% establecido en el criterio de aceptación se considera entonces que el modelo es pertinente, completo, adecuado y preciso, respecto a la evaluación llevada a cabo a los dos WS.**

Sobre la calidad de los WS, la estimación de los evaluadores arrojó un nivel de calidad **Intermedio** para el primer WS y **Avanzado** para el segundo, coincidieron en que el nivel de calidad especificado fue preciso, aunque en la categoría Eficiencia no proporcionó el nivel de precisión esperado por los evaluadores.

6. CONCLUSIONES

En esta investigación se propuso un Modelo de Especificación de la Calidad para Web Services, basado en características, con un total de 46 métricas. El principal aporte de este modelo es que parte de una base ya probada (MOSCA), el cual a su vez se inspira en estándares de calidad y propone métricas, para luego incorporar los aspectos propios de la calidad de WS y QoS, donde éste último es también una fuente de requerimientos importante íntimamente relacionados al mundo de la Ingeniería Web y las aplicaciones en Internet. El desarrollo de este modelo confirma que la calidad no es inflexible sino contextual. En este modelo se evaluaron las seis categorías: Funcionalidad, Eficiencia, Mantenibilidad, Fiabilidad, Usabilidad y Portabilidad. El estudio realizado a través de los dos WS arroja que al estimar la calidad de un WS se debe enfocar la misma en las categorías Eficiencia y Fiabilidad; considerando igualmente el aspecto de la Seguridad. No así la Usabilidad, la cual pierde relevancia para los Web Services por ser éstos una capa middleware con la que el usuario no actúa directamente. La evaluación del Modelo a través de dos WS, mostró que es pertinente a la calidad de los WS, completo en cuanto a su especificación, adecuado en el contexto de evaluación y preciso en el resultado alcanzado.

7. AGRADECIMIENTOS

Esta investigación fue financiada por el Fondo Nacional de Ciencia, Tecnología e Innovación (FONACIT) de la República Bolivariana de Venezuela, a través de los proyectos S1-2000000437 y S1-2001000792

8. REFERENCIAS

[1] Basili, V. y Weiss, D. A Methodology For Collecting Valid Software Engineering Data, *IEEE Trans. Software Eng. SE,* (1984) 10(6), pp. 728-738.

[2] Callaos, N. y Callaos, B., Designing with a systemic total quality, *Proceedings of the International Conference on Information Systems Analysis and Synthesis*, ISAS'96, (1996), pp. 15-23.

[3] Canta, C., Rufino, E., Restuccia, R. y Ruiz, F. *Web services*, Trabajo de investigación, Universidad Católica de Salta, Salta, Argentina, (2003).

[4] Dromey, G., A Model for Software Product Quality, *IEEE Transactions on Software Engineering* 2(2), February, (1995), pp. 146-162.

[5] Gillies, A. *Software Quality: Theory and Management*, Thomson, 2nd edition, (1997).

[6] Hangjung, Z. Designing Intra and Inter Organizational Business Processes with Web Services, Tampa, USA: *AMCIS. Ninth Americas Conference on Information Systems*, (2003).

[7] Hansen, G. Quality of Services, (1997), consultado el 05/05/2005 en: http://www.objs.com/survey/QoS.htm

[8] Humphrey, W. *Managing the Software Process*, Addison-Wesley Pub. Co., London, UK, (1998), pp.7.

[9] ISO/TEC 9126 1.2. Information Technology–Software Product Quality, Part 1, Quality Model, *ISO/IEC JTC1/SC7/WG6*,(1998).

[10] ISO/IEC TR 15504-2. Information Technology–Software Process Assessment, Part 2, A Reference Model Processes and Process Capability, Canada, *ISO/IEC JTC 1/SC 7*, (1998).

[11] Kitchenham, B., Jones, L Evaluation Software Engineering Methods and Tools. Part 6: Identifying and Scoring Features, *ACM Software Engineering Notes*, (1997), Vol. 22, No. 2.

[12] Losavio, F., Ortega, D. y Perez, M. Towards a Standard EAI Quality terminology, *Proceedings of XXIII International Conference of The Chilean Computer Science Society*, (2003) pp. 119-129.

[13] Maamar, Z. y Alkhatib, G. Integration of Web Services for Establishing Virtual Enterprise: An Agent-Based Perspectiva, Tampa, USA: *AMCIS. Ninth Americas Conference on Information Systems*, (2003).

[14] Mani, A. y Nagarajan, A. Understanding Quality of service for Web Services, (2002), consultado el 05/05/2005 en: http://www-106.ibm.com/developerworks/webservices/library/ws-quality.html

[15] Microsoft business solutions. E-business - *White paper*, (2003), consultado el 15/01/2004 en: http://www.dei.isep.ipp.pt/~qtdei/qtdei/MBSPT%20%20E-biz%20white%20paper.pdf

[16] Murray, M. An Inicial Investigation of Web Services in Healthcare, Kennesaw State University, Tampa, USA: AMCIS. *Ninth Americas Conference on Information System*, (2003).

[17] Mendoza, L., Pérez, A., Griman, A. y Rojas, T. Algoritmo para la Evaluación de la Calidad Sistémica del Software, *2das. Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, JIISIC 2002*, (2002) Brasil.

[18] Mendoza, L., Pérez, M. y Grimán, A. Prototipo de Modelo Sistémico de Calidad (MOSCA) del Software. *Computación y Sistemas Revista Iberoamericana de Computación*, (2005), Vol. VIII, No. 3, Mexico.

[19] Newcomer, E. *Understanding Web Services. XML, SOAP, and UDDI*, Boston, USA: Addison Wesley, (2002).

[20] Ortega, M, Pérez, M. y Rojas, T. Construction of a Systemic Quality Model for evaluating a Software Product. *Software Quality Journal*, (Julio 2003), 11(3), pp. 219-242.

[21] Perez, M., Rojas, T., Ortega, M., y Caldera, A. Toward Systemic Quality: Case study, 4Th Squad Meeting, Venezuela (1999).

[22] Pérez, L, M., Rojas, T., Mendoza, L. y Griman, A. Systemic Quality Model for System Development Process: Case Study, *Proceedings of AMCIS 2001*, (2001).

[23] Pérez, M., Mendoza, L. y Grimán, A. Hacia una Certificación de la Calidad Sistémica en los Sistemas de Software, *IV Congreso de Investigación y Creación Intelectual de la UNIMET*, (Mayo 2004), Caracas, Venezuela.

[24] Rincón, G., Mendoza, L. y Pérez, M. Guía para la Adaptación de un Modelo Genérico de Software, *IV Jornadas Iberoamericanas en Ingeniería del Software e Ingeniería del Conocimiento*, (Noviembre 2004), Madrid, España.

[25] Systinet Corp. Introduction to Web Services Architecture, consultado el 16/01/2004 en: www.systinet.com.

[26] Voas, J. Software Quality's Eight Greatest Myths, *IEEE Software*, 16(5), September/October, 1999, pp. 740-745.

Proposta de um Mecanismo de Desenvolvimento e Customização de uma Fábrica de Software Orientada a Domínios

José Augusto Fabri[1,2,3], Roberto C. Durscki[1], André L. P. Trindade[1,3,], Mauro de M. Spínola[1], Marcelo S. de Paula Pessôa[1]

[1]Departamento de Engenharia de Produção da Escola Politécnica da Universidade de São Paulo São Paulo – Brasil
[2]Centro de Pesquisas em Informática da Fundação Educacional de Assis – Assis-SP – Brasil
[3]Faculdade de Tecnologia de Ourinhos – FATEC – Ourinhos-SP – Brasil

{fabri@femanet.com.br, roberto.durscki@icet.pucpr.br, altrindd@yahoo.com.br, mauro.spinola@poli.usp.br, mpessoa@usp.br}

Abstract

This article has as objective to present a mechanism for the development of a software factory domains oriented. The conception of that mechanism appeared after to analyze the theories about software factory and software line-product. This analyzes culminates in the relationship of those twos concepts.

Keywords: Software Factory, Software Line-Product, Market Domain, Segment Market

Resumo

Este trabalho tem como objetivo apresentar um mecanismo de desenvolvimento e customização para fábrica de software orientada a domínios. A concepção desse mecanismo surgiu após uma ampla análise das teorias sobre fábrica de software e linha de produto de software, culminando na relação desses conceitos.

Palavras claves: Fábrica de Software, Linha de Produto de Software, Domínio e Segmento de Mercado.

1. INTRODUÇÃO

Atualmente, o mercado de desenvolvimento de software brasileiro trava uma batalha constante na busca de qualidade de produtividade. Essa informação pode ser comprovada ao analisarmos os vários programas de incentivo promovido pelo Ministério de Ciência e Tecnologia (MCT) através da Secretária de Política em Informática (SEPIN). Entre tais programas é possível destacar o SOFTEX (Sociedade para Promoção da Excelência do Software Brasileiro), cujos objetivos são: situar o Brasil entre os 5 (cinco) maiores produtores e exportadores de software do mundo e alcançar padrão internacional de qualidade e produtividade.

Além desses programas, o MCT e a SEPIN desenvolvem, periodicamente, uma pesquisa para verificar atributos de qualidade e produtividade do mercado brasileiro de desenvolvimento de software. A última delas foi publicada em 2002 e, ao efetuar uma análise nos dados é possível chegar às seguintes conclusões:

No Brasil existem cerca de 11.000 empresas com atividades relacionadas ao desenvolvimento e comercialização de software, essas empresas empregam 158.353 pessoas. Cerca de 25% das empresas possuem um programa de qualidade definido, outras 26% sentem a necessidade de estabelecer um programa de qualidade, isso comprova que o mercado brasileiro está tomando consciência em relação à necessidade da qualidade em seus produtos (contexto software) MCT-SEPIN (2002).

Paralelo a tais fatos existe uma discussão no âmbito empresarial e acadêmico sobre o tema "fábrica de software". Muitas empresas classificam o processo de desenvolvimento de software convencional como fabril. Porém, a maioria dessas empresas não possuem um processo que prime por produtividade e qualidade. Um processo de desenvolvimento que não atenda essas características, não pode ser considerado fabril, afirmação que pode ser constatada no trabalho desenvolvido por Costa (2003).

Costa (2003), apresenta uma pesquisa em 31 empresas, as mais significativas, que atuam no mercado brasileiro com a denominação Fábrica de Software, destas apenas 41% aplicam um ciclo completo de desenvolvimento de software para seus produtos; 45% aplicam metodologia própria; 16% utilizam ferramentas de controle de projetos; 14% possuem certificação CMM ou ISO; 13% utilizam ferramentas CASES e 10% aplicam métricas de qualidade.

Com base em tais números é possível afirmar que, apesar da tomada de consciência do mercado brasileiro em relação à produção de software, existe um grande trabalho a ser desenvolvido pela sociedade (empresas, universidades e governo) com a finalidade de atingir os objetivos estabelecidos pelo SOFTEX.

Nesse contexto, foi concebida a proposta de um mecanismo para o desenvolvimento e customização de uma fábrica de software orientada a domínios (segundo a teoria de linha de produto de software, domínio significa segmento de mercado). Com esse mecanismo é possível conceber uma fábrica de software, desde que a mesma tenha como pilar mercadológico (visão de mercado) o conceito de linha de produto de software.

Para atingir o objetivo proposto, esse trabalho foi estruturado em seções. A seção 2 apresenta a formalização dos conceitos de fábrica de software. A teoria de linha de produto de software é apresentada na seção 3. A seção 4 apresenta a relação entre os conceitos de fábrica de software e linha de produto de software, derivando o mecanismo de desenvolvimento e customização para fábrica de software orientada a domínios. Um caso real de utilização desse mecanismo pode ser verificado na seção 5. Por fim, a seção 6 apresenta as conclusões desse trabalho.

2. FORMALIZAÇÃO DO CONCEITO DE FÁBRICA DE SOFTWARE

Essa seção tem como objetivo apresentar as principais definições e características de uma fábrica de software.

Na literatura, é possível encontrar vários autores que trabalham, diretamente, com o conceito de fábrica de software, dos quais destacamos as principais definições.

Segundo Cusumano (1989), o termo fábrica de software foi utilizado pela primeira na década de 1960 no Japão. Várias empresas associam o termo ao mero desenvolvimento de software; entretanto, uma empresa produtora de software que não atende características como: produção em larga escala; padronização de tarefas; padronização de controle; divisão do trabalho; mecanização e automatização; não pode ser considerada uma fábrica de software. Para o autor, o desenvolvimento de uma fábrica de software implica nas boas práticas da engenharia de software aplicadas sistematicamente.

Já Bermer (1969) define fábrica de software como um ambiente no qual se constrói programas e se efetuam testes. Nesse ambiente devem existir ferramentas para realizar as ações de construir e testar. Uma fábrica deve possuir medidas de produtividade e qualidade, os registros financeiros devem ser mantidos por custo da programação e a forma de gerenciamento deve dar subsídios à previsão e à estimativa de dados de futuros projetos.

Para Basili et. al. (1992), uma organização com características de fábrica de software deve possuir uma estrutura de construção de software baseada em componentes. Os componentes utilizados podem ser desenvolvidos pela fábrica de componentes (ou unidade de produção de componentes). Através desta definição pode-se concluir que uma fábrica de componentes é a base para a implementação de uma fábrica de software.

Segundo Li et. al. (2001), uma fábrica de software deve possuir um conjunto de ferramentas padronizadas para construção de software, bases históricas para serem utilizadas no gerenciamento de projetos e, principalmente, um alto grau de reuso de código no processo de produção de um determinado software.

Fernstrom et. al. (1992) definem que uma organização fabril para o desenvolvimento de software deve estar calcada na questão "única do software", isto é, todo software é único, porém partes individuais são repetidas em vários projetos. O conceito fabril deve alicerçar o desenvolvimento, armazenamento e montagem das partes repetidas em um produto único.

Baseado nas definições apresentadas, esse trabalho encara fábrica de software como uma organização estruturada, voltada para a produção do produto software, totalmente alicerçada na engenharia e com forte caracterização pela organização do trabalho, pela capacidade de modularização de componentes e pela escalabilidade produtiva. Uma fábrica de software deve possuir um ambiente de gerenciamento de projetos; um processo padronizado, definido e institucionalizado; políticas que garantam a qualidade do produto software; um conjunto de ferramentas para mecanizar gerenciamento de projetos, processo e construção; técnicas para medir e estimar custo, prazo e tamanho de uma equipe para um determinado projeto; ambiente de teste definido e padronizado; foco em um segmento de mercado e; política de desenvolvimento de recursos humanos.

É importante salientar que existem vários modelos instanciáveis com o objetivo de configurar uma fábrica de software. A Tabela 1 apresenta algumas características presentes nesses modelos.

Através de uma analise na Tabela 1 é possível verificar as presença dos modelos C, D, E, F, G, H e I.

O modelo C foi definido por Cantone (1992) e contempla as seguintes características: processo de produção de software, processo de produção de componentes, base de componentes (idéia semelhante à biblioteca de componentes) e ferramentas (utilizadas na automatização do processo).

Características	**Definições e Modelos advindos da literatura**								
	A	B	C*	D*	E*	F*	G*	H*	I*
1. Unidade de produção de software.				•					•
2. Unidade de produção de componentes				•					•
3. Modelo de estrutura organizacional					•	•	•	•	
4. Modelo de organização da produção					•				•
5. Processo de produção de software	•	•	•	•	•	•	•	•	•
6. Processo de produção de componentes			•	•		•			•
7. Medidas e Estimativas		•				•			•
8. Gerência de Projeto de Software	•				•	•			•
9. Base de componentes			•	•	•				•
10. Base Histórica					•				
11. Qualidade		•	•		•	•			
12. Ferramentas	•	•	•	•	•		•	•	•
*Representa somente modelos.									

Tabela 1 – Características genéricas de uma fábrica de software.

O modelo proposto por Basili et. al. (1992), é caracterizado como modelo D e contempla as características da unidade de produção de componentes, responsável pela produção de componentes de código e de infra-estrutura (ativos de processo, formulários e *templates*); unidade de produção de software (produz software por meio da união e integração de componentes); processo de produção de software e componentes; e base de componentes.

O modelo E, proposto por Li et. al. (2001) atende à premissa que uma fábrica de software deve possuir um modelo de estrutura organizacional e um modelo de organização da produção, com características como: processo de desenvolvimento de software; gerência de projeto de software; bases históricas (informações de projetos já concluídos); base de componentes; qualidade de software e ferramental específico.

Fernandes e Teixeira (2004) propõem um modelo classificatório para fábrica de software, denominado nesse trabalho como modelo F, no qual é possível encontrar as seguintes características: modelo de estrutura organizacional; processo de produção de software e componentes; medidas e estimativas; gerência do projeto de software e; qualidade de software.

O modelo Eureka, nesse trabalho caracterizado como modelo G, surgiu com o projeto Eureka Software Factory descrito nos textos de Fernström (1991) e de Rockwell e Gera (1992). O objetivo do projeto é criar um mercado para produtos CASE, através de um consórcio que inclui 13 companhias de 5 países da Europa, tais companhias atuam nas seguintes áreas: manufatura de computadores, instituições de pesquisas, produção de ferramentas CASE e desenvolvimento de sistemas. Esse modelo possui as seguintes características: modelo de estrutura organizacional; processo de produção de software e ferramentas.

O modelo H, proposto por Bux e Marzano (1992), é caracterizado como uma biblioteca de processo de software pré-definida que suporta o conceito de fábrica de software. Essa biblioteca suporta características como: modelo de organização de produção; processo de produção de software e ferramentas.

Fabri et. al. (2004) propõem a configuração de um modelo de fábrica de software (denominado na Tabela 1 como modelo I) que abrange as seguintes características genéricas: unidade de produção de software e de componentes; modelo de organização da produção; processo de produção de software e de componentes; gerência de projetos; base de componentes e ferramentas.

As colunas *A* e *B* da Tabela 1 relacionam-se, respectivamente, como os conceitos propostos por Cusumano (1989) e Bermer (1969).

O conceito *A* mostra que uma fábrica de software deve possuir padronização de tarefas; padronização de controle e; divisão do trabalho, elementos ligados, diretamente, ao processo de software. A mecanização, ilustrada por Cusumano (1989), é um elemento que caracteriza a idéia de ferramenta apresentada na Tabela 1.

O conceito *B* evidencia a presença das características de processo de software (atividade de construir programas e efetuar testes); de medidas e estimativas e; de qualidade.

É importante destacar que as características genéricas foram mapeadas somente quando estas estavam explícitas nos modelos analisados. Se uma análise implícita for efetuada, certamente, será possível perceber que alguns modelos podem focar um número maior de características genéricas, porém, nesse trabalho, essa análise foi descartada, pois ao analisar Tabela 1, é possível perceber que todas as características levantadas estão presentes em, pelo menos, um dos modelos.

3. FORMALIZAÇÃO DO CONCEITO DE LINHA DE PRODUTO DE SOFTWARE

O termo linha de produto de software é uma adaptação do que vem sendo utilizado internacionalmente como *Software Product-Lines*. Esse termo é uma clara referência às linhas de produção da manufatura, as quais, no final do século XIX, introduziram uma revolução no processo produtivo, sugerindo o desenvolvimento seqüencial dos produtos, baseados em tarefas repetitivas, executadas sempre pelas mesmas pessoas que dispunham dos recursos materiais que necessitavam.

Segundo Cohen (2002), a definição de *software product-line* com maior aceitação na industria é a de Clements e Northrop que diz o seguinte:

> *"Uma linha de produto de software é um conjunto de sistemas que usam software intensivamente, compartilhando um conjunto de características comuns e gerenciadas, que satisfazem as necessidades de um segmento particular do mercado, e que são desenvolvidos a partir de um conjunto comum de ativos principais e de uma forma pré-estabelecida" Clements e Northrop (2002).*

Nessa definição, alguns termos devem ser destacados, pois representam as características mais importantes de uma linha de produto. Os termos são: *conjunto comum de ativos*, *forma pré-estabelecida* e *segmento particular de mercado*. A seguir, esses termos serão descritos, baseados em Clements e Northrop (2002).

Conjunto comuns de ativos: Os ativos são a essência da linha de produto de software e correspondem a um conjunto de elementos customizáveis, utilizados na construção dos softwares produzidos (os produtos). Entre esses ativos estão, por exemplo: componentes de software; modelos de documentos utilizados no processo de software (artefatos, ativos de processo ou componentes de infra-estrutura); *design-patterns* utilizados pela equipe de desenvolvimento; documentação dos requisitos comuns à família de produtos, a arquitetura de linha de produtos (que será a base da arquitetura de cada produto gerado) e; cronogramas. Dentre estes elementos, a arquitetura é chave e, normalmente, é estudada à parte dos outros ativos.

Forma pré-estabelecida: O processo de produção de softwares, através de uma linha de produto, é realizado por meio de planos de produção. Esses planos são definidos para cada software (produto) que será implementado pela linha. Ao definir um plano de produção, que dará origem a um novo produto, deve-se relacionar quais ativos farão parte deste produto e assim estabelecer um vínculo aos processos anexos a cada ativo utilizado. Os processos anexos são pequenos procedimentos internos de utilização que definem o que o ativo faz, qual a sua flexibilidade e qual a técnica de configuração do ativo (se ele for flexível). Essa natureza pré-estabelecida de funcionamento do processo produtivo da linha de produto de software é que garante o ganho de tempo e confiabilidade no desenvolvimento dos produtos.

Segmento particular: Muitas vezes chamados de domínio, refere-se ao corpo de conhecimento ou à área de especialização em que a linha de produto atua. O domínio está, diretamente, relacionado com o conjunto de funcionalidades correlacionadas que os produtos da linha pretendem atender. Como a flexibilidade dos ativos (especialmente da arquitetura de linha) é limitada, o modelo exige uma delimitação de um segmento de atuação. Sem essa delimitação, o escopo da linha de produto poderia ser muito mais abrangente, o que tornaria custoso criar e manter o conjunto comum de ativos.

Como se pode observar pela definição acima, esses conceitos foram, diretamente, derivados da indústria da manufatura, ou seja, derivam da filosofia de dividir para conquistar. Detalhando melhor, o processo sugere: dividir os produtos em pequenas partes, mais compreensíveis e gerenciáveis (ativo); possuir especificações claras e sem ambigüidades para produção / montagem (forma pré-estabelecida) e; atender a um domínio específico (segmento particular) para que, como em uma linha de montagem da indústria tradicional, cada novo produto seja uma reutilização de partes de produtos construídos anteriormente, pertencentes à mesma família, e, preferencialmente, com um mínimo de esforço adicional de construção.

4. A Relação entre Fábrica de Software e Linha de Produto de Software

Antes de apresentar a relação entre fábrica de software e linha de produto de software, os autores desse trabalho preocuparam-se em mostrar as principais diferenças entre esses conceitos.

4.1. Diferenças entre Fábrica de Software e Linha de Produto de Software

Analisando a seção 2, é possível verificar que o conceito de fábrica de software está preocupado com as seguintes questões:

- Configuração de uma infra-estrutura organizacional: Questão relacionada com as características estrutura organizacional e organização da produção;
- Definição do escopo de fornecimento de produtos da fábrica de software: A empresa é configurada como uma fábrica de programas somente, ou também é responsável por todas as atividades, desde o levantamento de requisitos até a manutenção do processo de desenvolvimento do produto?
- Definição do processo de produção de software: Quais serão as atividades e as tarefas inerentes ao processo, sabendo que uma fábrica *F* é configurada como fábrica de programas?
- Definição de um modelo de métricas para a produtividade. Questão ligada à característica medida e estimativa.
- Definição de uma infra-estrutura de armazenamento de componentes. Questão relacionada às características base de componentes e base histórica.

- Questões da qualidade e produtividade de software.

Já o conceito de linha de produto de software preocupa-se com:

- A relação dos produtos (software) imersos em um domínio, possibilitando que esses produtos sejam confeccionados a partir de uma arquitetura comum e de uma base de ativos compartilhados. A cuidadosa definição do escopo do domínio é fundamental para oferecer ganhos de custo na produção e assim sustentar a posição no mercado da entidade desenvolvedora de software.

- Questões relacionadas à customização dos ativos do processo (componentes de infra-estruturas ou artefatos do processo) e dos componentes de software, em relação aos domínios.

- A forma pré-estabelecida, que visa como definir ou customizar um plano produção para produzir um determinado produto da linha.

Enfim, uma fábrica de software provê o arcabouço conceitual para o aumento de qualidade e da produtividade em relação ao produto software. Já uma linha de produto de software trata de questões da aplicação dos conceitos fabris para software em domínios específicos de conhecimento e mercado.

4.2. Fábrica de Software e Linha de Produto de Software: Conceitos Diferentes que se Completam

Apresentadas às diferenças entre os conceitos de fábrica de software e linha de produto de software, uma analise torna possível perceber que a fábrica de software pode ser configurada para produzir produtos relacionados a um domínio específico (o conceito de linha). Essa afirmação pode ser visualizada por meio da Figura 1.

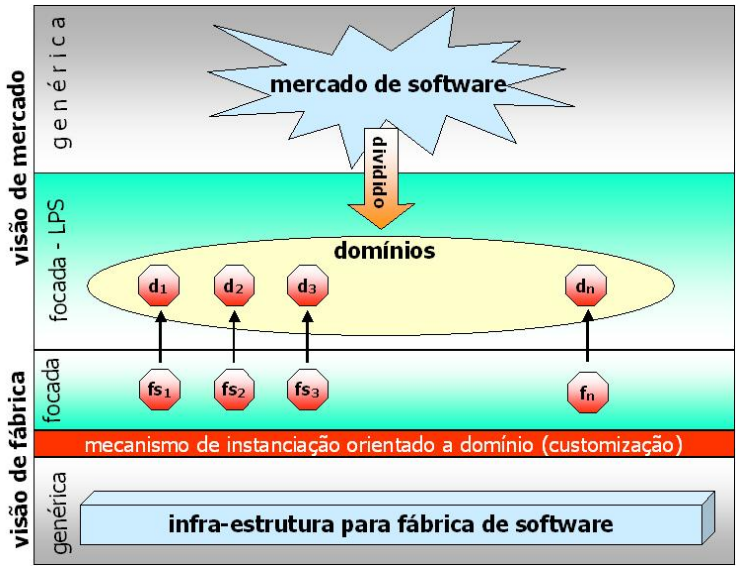


Figura 1 - Relação entre o conceito de fábrica de software e linha de produto de software

Analisando a Figura 1 é possível perceber que a visão de mercado pode ser configurada como genérica ou focada. Na visão genérica é possível olhar o mercado de desenvolvimento de software como um todo, isto é, enxergar todos os domínios e subdomínios de mercado. A visão focada está, intimamente, ligada ao conceito de linha de produto de software; nessa visão é possível perceber a questão dos domínios.

Conforme relatado na seção 3, o domínio está, diretamente, relacionado ao conjunto de funcionalidades correlacionadas que os produtos da linha pretendem atender e, ainda mais: o domínio exige uma delimitação do segmento de atuação de uma empresa de desenvolvimento de software.

O Ministério de Ciência e Tecnologia do Brasil (MCT-BR) apresenta uma delimitação de domínio de atuação das empresas de desenvolvimento de software, 70% das empresas brasileiras desenvolvem software sobre encomenda, 21%

desenvolvem o conceito de pacote de software para ser vendido no mercado, 7% desenvolvem software embarcado e 2% desenvolvem aplicações voltadas à Internet (dados extraídos MCT-SEPIN (2002)).

A delimitação de domínios apresentada pelo MCT-BR é bastante ampla sendo, assim, um bom ponto de partida para que uma empresa possa customizar uma fábrica de software para uma linha de produto de software, tendo em vista que esses domínios podem ser divididos em subdomínios.

É importante ressaltar que esse trabalho não se preocupou em apresentar uma configuração de domínios e subdomínios mais aprofundada.

A Figura 1 também apresenta a visão de fábrica, sendo que essa está dividida em visão focada e visão genérica. Dentro da visão genérica, é possível verificar a infra-estrutura de uma fábrica de software. Essa infra-estrutura deve atender as características genéricas apresentada na Tabela 1. A visão fabril focada é obtida após aplicação do mecanismo de instanciação orientado a domínio. A proposta desse mecanismo é customizar ou desenvolver uma fábrica de software para que a mesma possa atingir melhores índices de qualidade e produtividade para um domínio específico.

A Figura 2 apresenta, detalhadamente, o mecanismo de customização e possibilita verificar a presença da infra-estrutura para uma fábrica de software com as 12 características genéricas apresentadas e do conceito de domínio.

Para realizar a customização de uma fábrica de software orientada a domínio é necessário seguir os seguintes passos:

1. Definir o domínio de atuação da fábrica de software. A definição dos domínios configuram as bases histórica e de componentes – conjunto comuns de ativos.

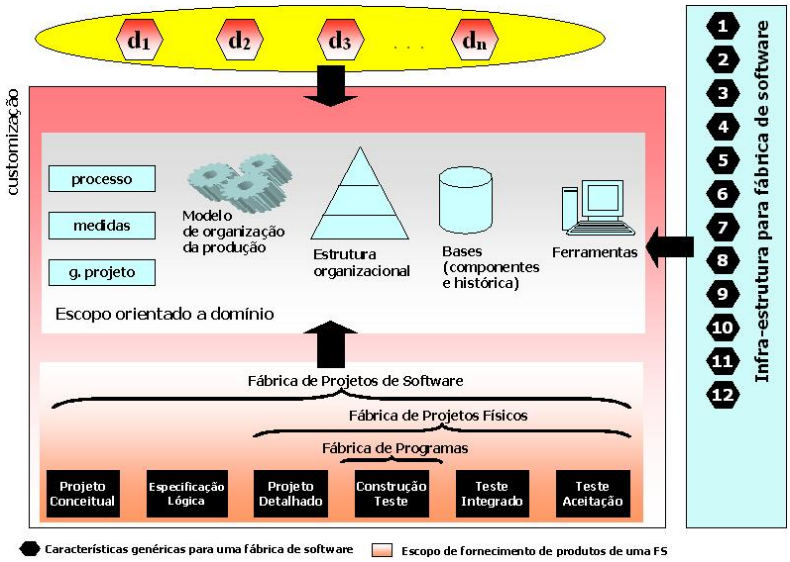


Figura 2 - Mecanismo de desenvolvimento e customização

2. Configurar o escopo de fornecimento de produtos de uma fábrica de software. Nessa configuração de escopo de fornecimento de produtos, Fernandes e Teixeira (2004) definem que uma fábrica de software pode ser classificada como:

   - Fábrica de projetos de software: abrange todo o ciclo de vida sistêmico para concepção do software (análise de sistema, projeto de software, construção, teste e implantação).

   - Fábrica de projetos físicos trata única e exclusivamente da idéia do software, abstraindo o modelo sistêmico, no qual o software está inserido.

   - Fábrica de programas, considerada a menor das entidades, tem como objetivo desenvolver componentes de código para a montagem do software. Essa entidade não está preocupada com o contexto sistêmico nem com o projeto físico do software, ela apenas produz código segundo a especificação advinda do projeto físico.

Salienta-se que configuração do escopo de fornecimento deve atender o domínio específico.

3. Definir a estrutura organizacional da fábrica: questões como organograma e a divisão de níveis organizacionais (estratégico, tático e operacional) são tratadas nesse passo.

4. Definir o modelo de organização de produção de uma fábrica de software. O modelo de organização de uma fábrica de software orientada a domínio deve possuir: uma unidade de engenharia de domínios, cuja responsabilidade é desenvolver os ativos de processo, por exemplo: *templates*, e formulários relacionados ao processo e; uma unidade de engenharia de aplicativos, cujas responsabilidades são definir customizar uma arquitetura genérica de produtos, instanciar componentes e definir o plano de produção.

5. Definir o processo de produção de software (quais as atividades e as tarefas inerentes ao processo de produção de software?).

6. Propor um conjunto comum de ativos (componentes de software, modelos de documentos utilizados no processo de software e *design-patterns*).

7. Configurar as medidas e estimativas a serem utilizadas na fábrica.

8. Configurar a forma de gerenciamento de projetos.

9. Configurar a base de componentes e a base histórica de projetos.

10. Definir um conjunto de ferramentas.

A união desses passos resultam no mecanismo de desenvolvimento e customização orientado a domínio (vide Figura 2).

5. Um Caso Real de Desenvolvimento de uma Fábrica de Software Utilizando o Mecanismo de Desenvolvimento e Customização Orientada a Domínios

O caso a ser descrito nessa seção pertence ao Centro de Pesquisas em Informática da Fundação Educacional do Município de Assis (CEPEIN-FEMA).

A Fundação é uma entidade sem fins lucrativos e mantém nove cursos de graduação, entre eles, cursos encontram-se o de Processamento de Dados e Bacharelado em Ciência da Computação.

O CEPEIN opera junto à FEMA e tem como objetivo fomentar estudos relacionados à área de Engenharia de Software. Atualmente, o CEPEIN conta com um corpo de cerca de 20 profissionais (professores, pesquisadores, projetistas, programadores e testadores) para a realização de suas atividades.
Em 2003, esse centro firmou uma parceria com o Departamento de Engenharia de Produção da Escola Politécnica da Universidade de São Paulo (DEP-POLI-USP) para o desenvolvimento de software para dispositivos pervasivos[1]. Segundo Burkhardt et. al. (2002), o termo pervasivo é empregado para abordar a façanha de distribuição, mobilidade e transparência. Em um cenário pervasivo, o usuário tem acesso à informação em qualquer dispositivo, por exemplo: um celular, um PDA ou um *desktop*.

Na parceria, o CEPEIN ficou responsável pela tarefa de codificação dos softwares. Isso levou o CEPEIN a projetar uma fábrica de programas orientada a domínios.

O projeto resultou no mecanismo de desenvolvimento e customização apresentado na Figura 2. Em um primeiro momento, o CEPEIN definiu o domínio de atuação: desenvolvimento de software para dispositivos pervasivos (passo 1, apresentado na seção 4.2). A justificativa para a escolha desse domínio está, intimamente, ligada a questões de mercado que, após realização de uma pesquisa, o CEPEIN verificou ser promissor, pois existem poucas empresas atuando nesse domínio. É importante ressaltar que a parceria entre DEP-POLI-USP e CEPEIN também contribuiu para a escolha desse domínio.

[1] O desenvolvimento de software para dispositivo pervasivo se constituí em um subdomínio dos domínios de desenvolvimento de software para internet e embarcado (passo 1 apresentado na seção 4).

Após essa definição, os responsáveis pela elaboração da parceria verificaram que seria necessário configurar uma fábrica de programa (passo 2), pois no momento da concepção da parceria, o CEPEIN ficou responsável somente pela codificação do software.

O passo 3 (seção 4.2) focou a necessidade de se definir a estrutura organizacional da fábrica de software. A estrutura utilizada pelo CEPEIN pode ser verificada na Figura 3, na qual é possível verificar as presenças: da missão da fábrica de programa perante o mercado de software; dos níveis estratégico, tático e operacional e; do organograma dos cargos em relação aos níveis.

No passo 4, o CEPEIN desenvolveu o modelo de organização da produção da fábrica de programas. As áreas de produção desenvolvidas pelo CEPEIN são: engenharia de domínio (definida no item 4 da seção 4); engenharia de aplicativos (definida no item 4 da seção 4); recepção de ativos de processo (artefatos relacionados a especificação do software); produção de código e; teste de software.

A área de recepção e customização de ativos de processo tem como objetivo verificar se os documentos da atividade de projeto de software (exemplo o documento de arquitetura), que chegam ao CEPEIN, estão seguindo o padrão pré-definido pelo Centro. Esse padrão tem como objetivos verificar se o documento de arquitetura do software possui todos os artefatos necessários para que a linha de produção dos programas possa ser instanciada. É necessário ressaltar que o documento de arquitetura do software não necessita retratar todas as funcionalidades, o mesmo pode apresentar partes delas, fato esse relacionado ao modelo de processo utilizado pela empresa contratante dos serviços do CEPEIN.

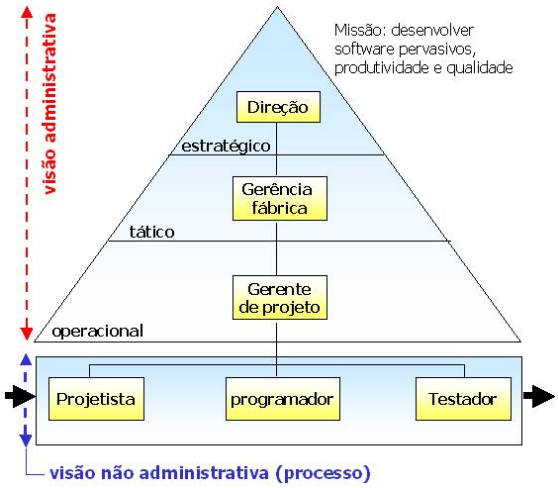


Figura 3 - Modelo organizacional da fábrica de programas do CEPEIN

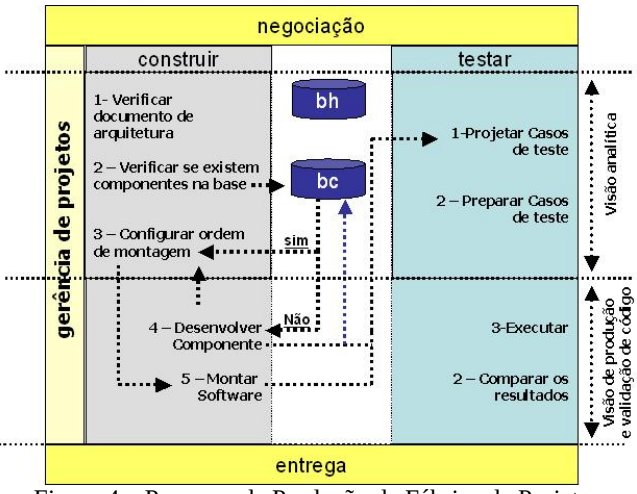


Figura 4 – Processo de Produção da Fábrica de Projetos (**bh** – base histórica de projetos e **bc** – base de componentes)

Ressalta-se que o CEPEIN não autorizou a divulgação do padrão citado no parágrafo anterior.

Após a definição do modelo de organização da produção, o CEPEIN definiu um processo de produção para a fábrica de programas (passo 5).

O processo de produção da fábrica de programas do CEPEIN pode ser verificado na Figura 4. No processo é possível verificar a presença de 5 atividades: Negociação, Gerenciamento de Projeto, Construção, Teste e Entrega.

Na atividade de negociação o diretor da fábrica de programas fecha os contratos para o desenvolvimento dos produtos. Essa atividade é, extremamente, importante, pois ela proporciona a relação direta entre a fábrica de programas e seus clientes (nesse caso as empresas de desenvolvimento de software que contratam a fábrica do CEPEIN para o desenvolvimento de componentes e softwares pervasivos).

A atividade de gerenciamento de projeto (passo 8) tem como objetivos planejar e controlar a execução do projeto dentro daquilo que foi planejado. Essa atividade possui as seguintes tarefas: definir o escopo do produto a ser gerado (no caso da fábrica de programas orientada a domínios, o escopo do produto deve respeitar a linha de produtos da fábrica); definir e seqüenciar atividades do projeto (essa definição deve respeitar as atividades do processo); definir cronograma, segundo as atividades, previamente, seqüenciadas na tarefa anterior e; estabelecer os recursos necessários para a execução do

projeto (recursos humanos, por exemplo). É importante salientar que a atividade de gerenciamento de projeto ocorre em paralelo às demais atividades do processo.

A atividade de construção possui as tarefas:

- verificar se o documento de arquitetura, advindo de algum dos possíveis clientes da fábrica de programas orientada a domínios, segue o padrão pré-estabelecido em contrato;

- verificar se os componentes existentes na base atendem os requisitos materializados no documento de arquitetura. Se atenderem, é necessário configurar a ordem de montagem (essa ordem expressa quais são os componentes necessários para que uma funcionalidade do software possa ser implementada). Caso contrário, o componente deve ser desenvolvido.

- Realizadas essas tarefas, o software entra em produção (tarefa montar software).

Na Figura 4 também é possível verificar que as tarefas desenvolver componentes e montar software disparam a atividade de teste, na qual os testadores realizam as seguintes tarefas:

- Projetar casos de teste: Segundo Sommerville (2003), os casos de testes são especificações das entradas e saídas esperadas do software. Os casos representam uma declaração do que está sendo testado;

- Preparar casos de teste: De posse das especificações, os testadores podem preparar os dados de testes que serão executados;

- Executar casos de teste: Execução dos componentes ou do software com os dados preparados;

- Comparar os resultados: Verifica se as saídas são condizentes com os casos projetados.

O teste, na fábrica de programas, pode ser realizado de várias formas, entre elas destaca-se o teste unitário (teste de um único componentes) e o teste de integração (vários componentes interligados para prover uma funcionalidade do software).

Por fim, na atividade de entrega é feito o teste de aceitação do produto pervasivo (software ou componente). Ressalta-se que nessa atividade, a empresa de desenvolvimento de software que terceirizou os serviços de fabricação de programas valida se o software ou componentes desenvolvidos estão de acordo com as necessidades relatadas no documento de arquitetura.

É possível perceber, na Figura 4, que a fábrica de programas desenvolvida no CEPEIN possui duas visões, analítica e produção e validação de código.

Na visão analítica, os envolvidos com o processo de produção analisam documentos, consultam a base de componentes, projetam casos de testes e preparam os dados para o teste. Nessa visão não é feito qualquer tipo de codificação.

Na visão de produção e validação de código os programadores e testadores desenvolvem componentes, montam softwares, executam os testes e comparam os resultados (validação de produto, segundo as especificações). Nessa visão é feita somente a codificação, segundo as especificações advindas da visão analítica.

O CEPEIN também configurou uma base de componentes (passos 6 e 9) com o objetivo de armazenar os ativos do processo. É importante destacar que todas as atividades utilizam esse conjunto de ativos (conceito advindo da teoria de linha de produto de software).

É importante salienta que a base de componentes é alimentada, constantemente, com novos ativos. Na Figura 4 é possível verificar o armazenamento de componentes pelo fluxo destacado em azul.

As medidas e estimativas utilizadas pelo CEPEIN (passo 7) estão baseadas em produção de componentes. Cabe ressaltar que o gerente de projeto tem total controle de sua equipe, e sabe quanto essa equipe é capaz de produzir em um determinado período. Os parâmetros utilizados pelo gerente de projeto estão armazenados na base histórica (passo 9) de projetos. Nessa base, os envolvidos com o processo de produção (projetistas, programadores e testadores) registram suas

atividades com uma periodicidade de 4 horas, registros esses chamados de "*logs*" de produção. Informações como data e hora de início de uma tarefa, evolução da tarefa e resultados alcançados, são armazenados nos *logs*.

Ferramenta	**Natureza**
Gerenciamento de Projetos.	Software Livre
Armazenamento e Recuperação de Dados das Bases.	Produzida pelo CEPEIN
Gerenciamento de *Logs* de Produção	Produzida pelo CEPEIN
Ferramenta de Diagramação e Modelagem	Software Livre
IDE para Desenvolvimento de Código	Software Livre
Ferramenta para Calculo de Medidas e Estimativa	Produzida pelo CEPEIN
Ferramenta para Controle de Versão	Software Livre
Ferramenta para Gestão de Requisitos	Produzida pelo CEPEIN.

Tabela 2 – Conjunto de Ferramentas Utilizadas pela Fábrica de Programas do CEPEIN

A fábrica de programas configurada no CEPEIN utiliza um conjunto de ferramentas (passo 10) para automatizar o processo de produção e agilizar o acesso às bases (componentes e histórica). O conjunto de ferramentas é apresentado na Tabela 2.

Na Tabela 2 é possível verificar que as ferramentas são classificadas como software livre ou como produzidas pelo CEPEIN. É importante ressaltar que as ferramentas produzidas pelo CEPEIN foram materializadas antes da configuração da fábrica de programas em 2003.

Um dos Produtos Gerados pela Fábrica de Programa

O software ManWapp, constituí-se como o principal produto da linha. Este software tem como objetivo automatizar a troca das informações entre os prestadores de serviços de manutenção de telefonia, de rede elétrica, de água e esgoto e as empresas que utilizam esse tipo de mão de obra (por exemplo: prestadora de serviço de telefonia, distribuidora e energia e água) e os clientes dessas empresas.

Salienta-se que ficou sob responsabilidade do CEPEIN somente a fabricação dos componentes de código e a montagem desses componentes para constituir o software.

A parte da arquitetura ficou sob a responsabilidade do Departamento de Engenharia de Produção da Universidade de São Paulo (vide arquitetura na Figura 5).

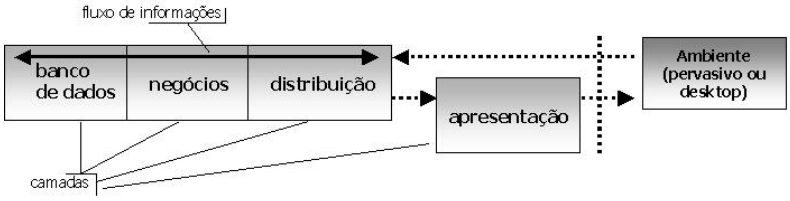


Figura 5 – Visão Arquitetônico do ManWApp

Nessa figura, é possível perceber que o software está dividido em 4 camadas: banco de dados, negócio, distribuição e apresentação.

A característica de "pervasividade" pode ser verificada nas camadas de distribuição e apresentação. A camada de distribuição detecta qual dispositivo está solicitando as informações armazenadas na camada de dados, os dados são manipulados pela camada de negócios, fica sob responsabilidade da camada de apresentação formatar, automaticamente, as configurações para exibição das informações em tais dispositivos.

6. CONCLUSÕES E TRABALHOS FUTUROS

Esse trabalho apresentou a proposta de um mecanismo para desenvolvimento e customização de uma fábrica de software orientada a domínio. Salienta-se que o mecanismo foi utilizado para desenvolver a fábrica de software (escopo de fornecimento classificado como fábrica de programas) do CEPEIN.

Um fator interessante verificado no trabalho é a presença das bases (componentes e históricas). Nessas bases são armazenados os ativos do processo de desenvolvimento de software, componentes de código e dados históricos dos

projetos. Todas as informações armazenadas nas bases convergem para um domínio especifico, solidificando assim o conceito de orientação a domínio proposto no trabalho.

Um exemplo típico de orientação a domínio da fábrica desenvolvida pelo CEPEIN é verificado na seção 5. Nessa seção é apresentada a arquitetura do principal produto desenvolvido pelo Centro (software ManWapp). Os componentes de código gerados para materializar a arquitetura sustentam toda a linha de produção do CEPEIN. Conclusão essa obtida após verificar que maioria dos softwares pervasivos devem possuir a arquitetura de 4 camadas apresentada na Figura 5.

Um dos principais problemas enfrentados para instanciar a Fábrica de Software orientada a domínios foi à mudança cultural dos envolvidos no processo e a parametrização para que os componentes de código pudessem ser reutilizados.

O mecanismo de customização e desenvolvimento de uma fábrica de software apresentado nesse trabalho possui duas visões, a orientada a domínios, discutida no decorrer desse trabalho, e a visão de configuração de escopo. Essa última determina se a fábrica de software será configurada como fábrica de projetos de software; fábrica de projetos físicos ou; fábrica de programas (vide figura 2)

Até o momento o mecanismo de desenvolvimento e customização proposto não foi aplicado no desenvolvimento ou na customização de uma fábrica de projetos físicos e em uma fábrica de projetos de software, fator esse que constituí em um ponto fraco desse trabalho.

Como trabalho futuro, seria interessante aplicar o mecanismo para desenvolver ou customizar uma fábrica de projetos físicos para um domínio específico.

7. REFERÊNCIAS

[1] Basili, V. R.; Caldiera, G.; Cantone, G.; A Reference Archiecture for the Component Factory. *ACM Transaction on Software Engineering and Methodology*. Vol 1. nº 1. pp 53-80. January 1992.

[2] Bermer, R. W. The Economics of Program Production**.** *In Information Processing 68*, North-Holland, Amsterdan. 1969.

[3] Burkhardt, J. et. al. *Pervasive Computing – Technology and Architecture of Mobile Internet Application*. Addison Wesley. 2002.

[4] Bux, G.; Marzno, G. Libary of Predefined Software Process Models as Support for Software Factory: Design the SFINX Proposal. *IEEE Software*. March 1992.

[5] Cantone, G. Software Factory: Modeling the Improvement. *Competitive Performance through Advanced Technology. Third International Conference* (Conf. Publ. No. 359). Pages: 124 – 129. 27-29. 1992.

[6] Clements, Paul; Northrop, Linda. *Software Product Line: Practices and Partterns. Boston*. Addison Wesley, 2002.

[7] Cohen, Sholom. *Product Line State of the Practice Report*. Disponível em http://www.sei.cmu.edu/publication/document/02.reports/02tn01.html, 08/07/2004.

[8] Costa, Ivanir. *Contribuição para o aumento da qualidade e produtividade de uma fábrica de software através da padronização do processo de recebimento de serviços de construção de softwares*. 174 p. Tese (Doutorado). Departamento de Engenharia de Produção. Universidade de São Paulo. 2003.

[9] Cusumano, Michael A. Software Factory: A Historical Interpretation. *IEEE Software*. Vol. 6, No. 2. pp. 23-30. March/April 1989.

[10] Fabri, José Augusto; Trindade, André Luiz Presende; Begosso, Luiz Ricardo; L'Erário, Alexandre; Silveira, Fábio Luiz Fuji; Pessôa Marcelo S. de Paula. Techniques for the Development of a Software Factory: Case CEPEIN-FEMA. *Annals of 17th International Conference Software & Systems Engineering and their Applications*. Paris. December 3, 2004.

[11] Fernandes, Aguinaldo Aragon; Teixeira, Descartes de Souza. Fábrica de Software: *Implementação e Gestão de Operações*. Editora Atlas. 2004.

[12] Fernström, C. The Eureka Software Factory: *Concepts and Accomplishment. Proceedings Third European Software Engineering Conference*. Berlin 1991.

[13] Li, C.; Li, H.; Li, M. A Software Factory Model Based on ISO 9000 e CMM for Chinese Small Organization. *Second Asia-Pacific Conference on Quality Software (APAQS'01)*. Hong Kong. December, 2001.

[14] MCT-SEPIN. Ministério da Ciência e Tecnologia da Secretária de Política de Informática. *Relatório de Qualidade e Produtividade no Setor de Software Brasileiro N4* - ISSN 1518-112X, 2002.

[15] Rockwell, R.; Gera, M. H. The Eureka Software Factory CoRe: A Conceptual Reference Model for Software Factories. *Software Engineering Environments Conference Proceedings*. Pages:80 – 93. July 1993.

[16] Sommerville Ian. *Engenharia de Software*. 6ª Edição. Addison Wesley, 2003

A Measurement-based Approach for Improving Software Practices

Liliana K. Guzmán
Universidad de Valparaíso, Departamento de Computación,
Valparaiso, Chile
liliana.guzman@uv.cl

and

Marcello A. Visconti
Universidad Técnica Federico Santa María, Departamento de Informática,
Valparaíso, Chile
visconti@inf.utfsm.cl

Abstract

In recent years an increasing number of software organizations have launched initiatives to improve their software process. Unfortunately, most of them have been unable to move beyond diagnosis and action planning, turning those plans into real and practical actions. This article focuses on using an adaptation of SQUID (Software QUality In the Development Process) method – a measurement-based methodology for specifying, monitoring and evaluating the software product quality during development - to manage the quality of implanting generic tools for improving specific software practices. We analyze the rationale for using this approach, describe the initial implantation process designed and the preliminary applications conducted that led to the formulation of an improved version for the implantation process, and finally show how the proposed adaptation helps in formalizing and normalizing the implantation process, setting tangible goals and evaluating the results more precisely.

Keywords: measurement, adoption process, software practice

1. INTRODUCTION

Over the years, the efforts of software engineering for achieving high quality products, developed in time and within budget, have been based on the premise that process quality affects product quality. Based on this premise, several initiatives have been launched that try to improve productivity, product quality and customer satisfaction through Software Process Improvement (SPI) [3].

Based on this approach, many software organizations have begun SPI programs that include technology transfer, process modelling and training based on process improvement. In most cases SPI programs are supported during the assessment and planning by specific models or standards like CMM, CMMI, SPICE, BOOTSTRAP, PSP, TSP, ISO 9000, etc [3],[5],[6],[7]. Then organizations strive with great difficulty to move past the diagnosing and establishing phases into real action (Acting Phase in the IDEAL Model). The mayor cause seems to be the absence of guidance in transforming findings and recommendations into improvement actions [3],[6].

From an organizational and practical view, one approach to solve this problem has been to adopt a set of generic tools, such as procedures, guides, templates and checklists, to facilitate and reduce the effort of implanting a specific practice, such as Software Quality Assurance or Risk Management. A successful case is the Essential Set product developed by Software Process Center. The generic tools mentioned above don't substitute an SPI assessment and plan, they only support a specific practice adoption. At the Acting phase the adoption process of these generic tools is a critical factor for the success of SPI projects related to a specific practice, not only for obtaining a high quality final product, in this case the practice adoption, but also for the perception of the practice and SPI initiative contribution.

Technology adoption is a complex process, especially when a new practice is involved. Like any other project, it involves organizational commitment, goals, planning, products, milestones, resources, management, measurement, etc.

At the same time it must also deal with multiple dimensions, different audiences, several levels of diffusion and use, transition mechanisms and with a nonlinear process [8]. If we add the lack of adoption process frameworks to support this kind of projects, the challenge seems to be even more complex.

In a previous paper [9] we introduced a first version of an adoption process model for implanting specific practices through a set of generic tools, such as procedures, guides, templates and checklists. Based on this process, we also established a measurement-based approach to manage the quality of adopting specific tools. The results of a partial application showed this approach as a sound and useful scheme for the adoption process and for the definition and evaluation of the quality requirements of the tools finally adopted. It provides greater consistency and formality to the adoption process, and greater visibility and process understanding to the participants. All these contribute to move past the assessment and planning of SPI programs into real action and to improve the perception of SPI initiatives inside organizations. In this paper we present a refined version of the measurement-based approach. The refinements take into account our experience in applying the earlier version into two case studies in the Chilean software industry.

This paper is organized as follows: we present the initial proposal for a measurement-based approach for the software practice adoption process, discuss the main findings and lessons learned after applying the adoption process in two different scenarios, and end with a proposal for an improved measurement-based approach for the adoption process of software practices.

2 Development of a measurement-based approach for implanting software practices

The measurement-based approach for adopting generic tools has been developed based on SQUID (Software QUality In the Development Process) [1],[4]. SQUID is a measurement-based methodology and tool for managing the quality of a software product during its development, whose purpose is to assure the delivery of a product according to the quality requirements defined initially. Its goal is to support effectively the control and evaluation of quality for a software product, and to allow the organization to conduct and analyze quality measurements according to its own institutional reality.

SQUID methodology defines quality as the operational behaviour of a product required by users, i.e. it associates the quality to a set of product properties. These properties are made up of external or internal product characteristics that must be broken down into subcharacteristics up to the establishment of measurable attributes. The external properties correspond to the way the product operates, while the internal ones to the way the product is being developed. Based on this definition of quality, SQUID assumes that the internal product characteristics influence their external characteristics, so it suggests the definition of quality requirements through the specification of target values for the external characteristics of the product, and proposes to reach those objectives by defining and monitoring the target values of the internal characteristics.

The SQUID methodology proposes a quality process that includes the following activities:

- Specification of quality requirements for the product in terms of specific target values for the external characteristics and the viability assessment of those requirements before starting the project.
- Quality planning by selecting a development model and defining target values for the internal characteristics.
- Quality control through the use of internal measurements. This is, progress monitoring towards quantitative requirements during development, measuring the internal properties and making the proper adjustments.
- Quality evaluation of the end product by comparison of the specified target values for the external with the actual values.

The SQUID tool supports all these activities as they make up its conceptual model. Nevertheless, to begin the process it is necessary to configure SQUID data base, which requires the definition of development, quality and measurement models, and their integration, as explained below.

- *Development model.* For each project the model will be characterized by objects types that are classified as: deliverables generated during the development, activities that produce deliverables, and review points. This definition provides consistency to the measurements and allows using information about past projects to assess the viability of current and future projects. Quality measurements will be conducted on object types at specific review points.

- *Quality model.* The quality model will be specified in terms of quality characteristics that must be broken down up to the definition of attributes that are measurable directly. These characteristics and attributes can be internal or external.
- *Measurement model.* The measures collected through SQUID can be actual, estimate or target. An actual is the current value measured for specific quality attribute. An estimate can be obtained from past data or through the SQUID toolset's estimation facilities. Target values are specified during the quality specification phase. The SQUID user defines how to take measurements in counting rules, which clarify and regulate all the conditions that could affect the measures objectivity.
- *Models integration.* All internal attributes belonging to specified quality model must be allocated to project object types belonging to specified development process and associated to units and counting rules.

Focusing on how SQUID models are described, we realize that they can be defined for any productive line. The product does not need to be necessarily software. Furthermore, the specification, planning, control and evaluation of quality is applicable to the development of any product. Thus, it is reasonable to think that SQUID is flexible enough to be used to manage the quality of developing any product.

This research adapted and configured SQUID to be used as a tool for the quantitative management of the quality of a process of adopting a set of generic tools for specific software practices. The product corresponds to a set of generic tools that must be implanted and the process corresponds to the adoption of these tools.

4 FIRST VERSION OF ADOPTION PROCESS

In this section, we describe the first version of the measurement-based approach in terms of the SQUID models. For more details see [9]. Note that the configuration is only a generic proposal, its real application requires adjustment to the goals of a specific adoption.

4.1 Development model: An adoption process model

The first version of the generic tools adoption process is shown in Table 1. Assuming organizational commitment, an assessment of current practice, an action plan and the assignment of a person responsible for the adoption process, the model includes five phases: a project and tool preparation phase, a pilot test for its usage, an analysis of the pilot results, the refinement required to complete its institutionalization, and the tool institutionalization itself.

Table 1. First version of the adoption process model

Phase	**Object Types**		
	Activities	**Review points**	**Deliverables**
Preparation	Analysis of generic tools to adopt	Specification approval	Specification of generic tools to adopt
	Tools adaptation	Preliminary generic tools version approval	Preliminary tools version
Pilot Test	Projects selection	End of testing	Preliminary version of used tools
	Orientation		Test reports
	Tools incorporation into project		
	Test execution		
Analysis of results	Analysis of results	Results report approval	Results report
Refinement	Tools refinement	Official tools version approval	Official tools version
Institutionalization	Selection of a responsible team	Training plan approval	Training plan
	Training planning		
	Training	End of implantation	Implanted tools

4.2 Quality Model

After the configuration of the Development Model, we establish the quality characteristics that make up the Quality Model. Those characteristics are broken down in subcharacteristics and attributes and are classified as external or internal, as shown in Table 2. Note that for this version of the adoption process model the internal and external view are identical.

Table 2. First version of the quality model

Internal and External View		
Characteristics	**Subcharacteristics**	**Attributes**
Functionality	Fitness	satisfaction, usefulness
	Completeness	omissions, remaining, coverage
	Consistence	errors
	Non ambiguity	legibility, standardization, uniformity
Usability	Understandability	time, complexity, legibility, standardization
	Learnability	time, complexity, legibility, standardization
Operability		adherence, compliance, correctness
Generality	Adaptability	degree of process independence
	Interoperability	compatibility
Cost		effort, time
Maintainability	Analyzability	time, size, complexity
	Changeability	time, size, complexity
	Testability	time, coverage
	Modularity	coupling, cohesion

4.3 Measurement Model

The Measurement Model specifies how the external attribute must be collected. Measurements are expressed in different units and have a counting rule associated, that clarifies and regulates the conditions under which the measurement can objectively be made. For an example see Table 3. The real application of the model needs to define the measurements as target, estimated and actual value.

Table 3. Example of measurement model

Characteristics	**Subcharacteristics**	**Attributes**	**Counting Rule**
Functionality	Fitness	satisfaction	*on/off* survey for the implantators and/or developers
	Completeness	coverage	(# covered objectives)/ (# defined objectives)
Cost		effort	average man hours average required to use the tool
		time	days required to use the tool

4.4 Models integration

Once the development model and internal view of quality model have been defined, they are linked into a collated model. During the collation process all the internal attributes are associated to objects types of the development model, assigning a unit and a counting rule to each internal attribute. For an example see Table 4.

Table 4. Example of models integration

Object Type (deliverable): Specification of generic tools to adopt			
Characteristics	**Subcharacteristics**	**Attributes**	**Counting Rule**
Usability	Understandability	time	average time required by the implantators and/or developers to understand the purpose and the possibilities of using the product
		complexity	effort / # functions
		time	days

5 PRELIMINARY APPLICATION OF ADOPTION PROCESS

The first version of the measurement-based approach for implanting specific software practice has been used on two software practice adoption projects: Formal Technical Review (FTR) and Defects Analysis. Both experiences took place in a medium size development company located in Valparaíso, Chile.

5.1 Experience 1: Adoption of formal technical review

The first project used the proposal to manage part of the implantation process of a set of generic tools related to FTR process. This application focused on configuring the generic proposal, establishing target values for the quality attributes for all phases of the adoption process and getting actual values for the preparation phase.

At the configuration activity, the development model proposed for the tool adoption process was used unchanged because of its completeness and adequacy for the situation at hand. The quality, measurement and integration models were adapted from generic models considering organizational requirements. The specification of target values for the quality attributes was based solely on the participant's experience. The organization didn't have historical data to facilitate this task. Then, the preparation phase started. It consisted of a study and selection of the tools to be adopted, and adaptation of the tools to the organizational requirements. Full results of this first application experience can be found in [9].

5.2 Experience 2: Adoption of defect analysis

The second experience is related to the adoption of the Defect Analysis practice. In this case the proposal was used unchanged to manage the adoption process in all its phases and the specification of the target values was based on participant's experience. In spite of the recommendations outlined in section 2, the project started without an assessment of current practice and plan action.

The preparation phase was focused in defining the organizational requirements associated to the defect analysis process from a tactical management viewpoint. It also specified and established the preliminary generic tools version. Then three software development projects were selected for the pilot test. At the beginning of the pilot test, the projects were in different life cycle phases. Therefore it wasn't possible to assign proper time and efforts to defect analysis activities. These activities were developed on very hard conditions that sometimes put in risk the adoption project. During pilot tests, new requirements arose. To satisfy these needs, analysis and refinement phases were performed in parallel. After the pilot tests, stakeholders analyzed the impact of the defect analysis process and the quality of adoption process. This analysis led to the approval of the generic tools version without changes and authorized the institutionalization of the defect analysis process. The results of each phase were informed to the entire organization. This action gave more visibility to the project and increased the organizational commitment. Finally the organization entered the institutionalization phase. Full details of this experience can be found in [7].

5.3 Lessons learned from preliminary applications

These experiences showed the company the need to formalize the adoption process. Furthermore, defining target values facilitated the specification of precise and tangible expectations in relation to the generic tools adoption, which directly leads to a greater understanding of the purpose of implanting them, and to a more objective evaluation of the final results. It was also possible to observe that the generic models make up a good starting point for the particular configuration of the organization under study. In global terms, the proposal properly supports the adoption process of software practices. It was useful and effective.

Nevertheless, we observed a number of lessons learned that were instrumental in the proposal of a second version for the tool adoption process, namely:

- *The defined input conditions are necessary but not enough.* Several experiences of SPI initiatives in small organizations indicate that market demand is the main reason to start SPI programs. When facing a global market, small organizations find themselves under a great pressure to achieve some kind of certification, such as CMMI or ISO. Therefore many of them began multiple practice adoption programs without an appropriate diagnosis and SPI Plan, so they don't assess amount of change they can reasonably absorb and don't adjust expectations and schedules accordingly, leading to potential failures. The second case study is an example of this circumstance.
- *The development model is ambiguous and incomplete.* The first version was defined in terms of activities, review points and deliverables. This approach is useful from SQUID methodology view, although it isn't enough for people who must plan and participate on the adoption process. This situation arose at the beginning of the second experience. For example, participants asked for: Who should participate on activities? Which are their responsibilities? Who must be informed about what? Which is the minimal table of contents of the deliverables?

Which are the criteria to close pilot tests? These questions show that the first version of the adoption process needed to include further detail.

- *An adoption plan is needed.* Each adoption process makes up an important initiative to the organizations that need resources, scheduling and a project plan. The action plan exists only in a few cases and almost always is too general.
- *The diffusion of the adoption process progress is critical to increase organizations commitments.* Diffusion activities motivate and maintain informed the whole organization about initiatives objectives and results. The experience show that increasing the project visibility increases the understanding of the initiative importance, myths fall down, increases the organizational commitment and the later institutionalization is favored.
- *Measurements must be used to be effective.* Both study cases show that measurements contributes to make decisions and to the project visibility. Nevertheless, a number of measures were not possible to collect. This meant an additional and useless effort that put in risk the measurement program. Measurement must help decision makers understand project and organizational issues. Try to measure so much can collapse the measurement activities: people lose focus on what is being measured and why.
- *Define measurement terms to provide everyone working in the project a common understanding of how the attributes must be collected, what is being measured, and how this relates to the decision making.* The collection of actual values of each measurement shouldn't depend on individual criteria. Each measure must be needed, non ambiguous, objective and repeatable. Only in this way, they will be useful for decision makers. Therefore the description of counting rules and units for each attribute must be explicit.
- *Decisions makers must understand what is being measured.* Without accurate information, the managers were not able to properly make decisions about a specific adoption practice. Stakeholders need early information to reduce risk and correct problems. In this context the quality and measurement model and the models integration are a good starting point.

6 SECOND VERSION OF ADOPTION PROCESS

The second version was the result of the analysis of both case studies, new work sessions with qualified engineering associated to SPI initiatives and an iterative review process of experiences and recommendations for better technology adoption practice and measurement programs [6],[8]. Main results are summarized next:

1. Stress the entry conditions to assure the proper adoption project kick off.
2. Describe the adoption process taking into account the SQUID development view and the ETVX approach [2]. Second version of adoption process model cover the following aspects: activities and process phases, activities network, objectives, roles and responsibilities, input/exit criteria, commitments, review points and deliverables.
3. Include an adoption project plan.
4. Include a period of tool use during the institutionalization. This activity allows quantifying the impact of the practice/generic tools adoption on the development process.
5. Include a information phase across the process. This supports project visibility and increases organizational commitment.
6. Specify a flexible and effective measurement program. That is, limit internal and external measurements only to those whose results are good, i.e they measure what is intended, and are effective in supporting decision making.
7. Improve counting rules and measurement units so ambiguities are left out both in metrics collection and interpretation
8. Include simple associations between measured attributes and decisions to be taken.

6.1 Development process: An improved adoption process

The second version of the generic tools adoption process is shown in Figures 1 and 2. The model include seven phases: a project planning and practice/tools preparation phase; an orientation phase to introduce, motivate and train pilot test participants on practice/tools; a pilot test for its usage, an analysis of the pilot results, a refinement phase to establish the official practice policy and the official tools version, the institutionalization and a diffusion phase to promote project results and to assure organizational commitment. In order to reach the process the model considers the following key aspects:

- *Organizational commitment.* Adopting a practice affects the entire organization. If senior management don't lead and haven't a permanent participation on the initiative, the remainder organization will underestimate and

potentially leave it. To set down this condition, the new adoption process model includes for each phase an organizational commitment section.

- *Assign a full time person to the SPI program.* SPI is a continuous and complex process. It needs permanent management and leadership, and a software process educated staff. At least, organizations must assign a full time person. With regard to adoption process management, the model establishes an Adoption Process Responsible (APR) as a permanent role along the process.
- *Define explicit objectives and their relationship with the organizational goals.* It is correct to start a SPI program from a market request. However, an effective SPI program should be based on organizational goals and needs. Any SPI initiative, including generic tools adoption, must define explicit objectives and their relationship with the organizational goals. This is the essential criterion for making decision during any project.
- *Set the baseline of the organization current state.* To establish the organizational requirements, an effective solution and the impact of this solution on the development process, its baseline current state must been known. Otherwise, the solution might be incorrect in its origin.
- *Analyze the practice adoption impact.* Best practices adoption affects all task and resources, specially people, at tactical and technical organization levels. Therefore it's important to analyze the adoption practice impact looking at benefits and potential risks. The model assures this condition considering its results as a part of the specification of practice and generic tools to be adopted, it's a deliverable of the preparation phase.
- *Quantify expected results.* Organizations always have expectations about practice adoption: increase productivity, increase quality, reduce risk, reduce cost, early defect detection, etc. Short and long-term expected results should ne quantified so that adoption results can be better evaluated. Otherwise senior management could dismiss the initiative, before benefits can be made visible.

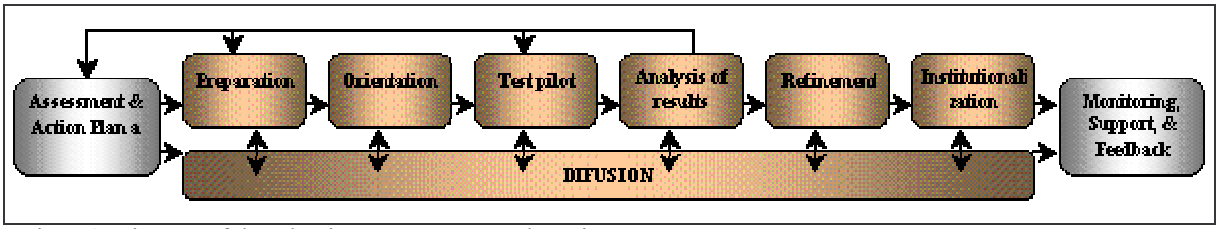
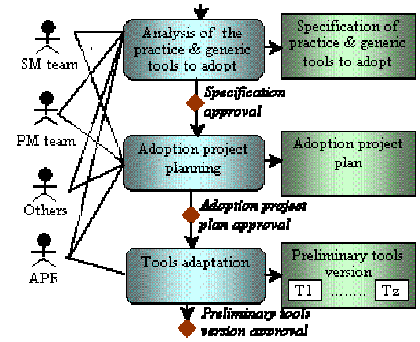
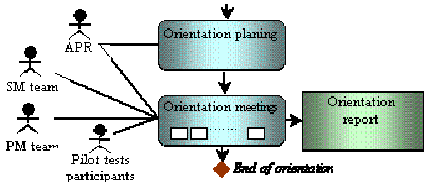


Figure1. Diagram of the adoption process - second version.



PREPARATION

Input criteria: Organization commitment. Assignment of an APR. Establishment of explicit objectives and their relationship with the organization goals, the baseline of organization's current state and target objectives. Analysis of the practice/generic tools adoption impact. Objective: To analyze the practice/generic tools. To establish an appropriate solution. Exit criteria: Adoption project plan and preliminary tools version are established and approved. The adoption project plan established pilot tests projects and audience. Commitment: All participants commit time and effort required to complete the phase. SM's commit to allow the APR to explore and develop an practice adoption proposal, to provide APR with business need for the practice adoption and with the resources necessary to develop the proposal. The APR must regularly communicate results of its work to the entire organization..



ORIENTATION

Input criteria: Selection of appropriate projects and audience for pilot tests. Adoption project plan and preliminary tools version approval. Objective: To introduce, motivate and train pilot test participants on practice/preliminary tools version. Exit criteria: The orientation meetings have covered all pilot tests participants and all level of training: knowledge, use and support. Commitment: All participants commit time, effort and training required to complete the phase. SMs must convince people that management is serious about the initiatives. .SMs and PMs should learn and use the same tools that the remainder participants.

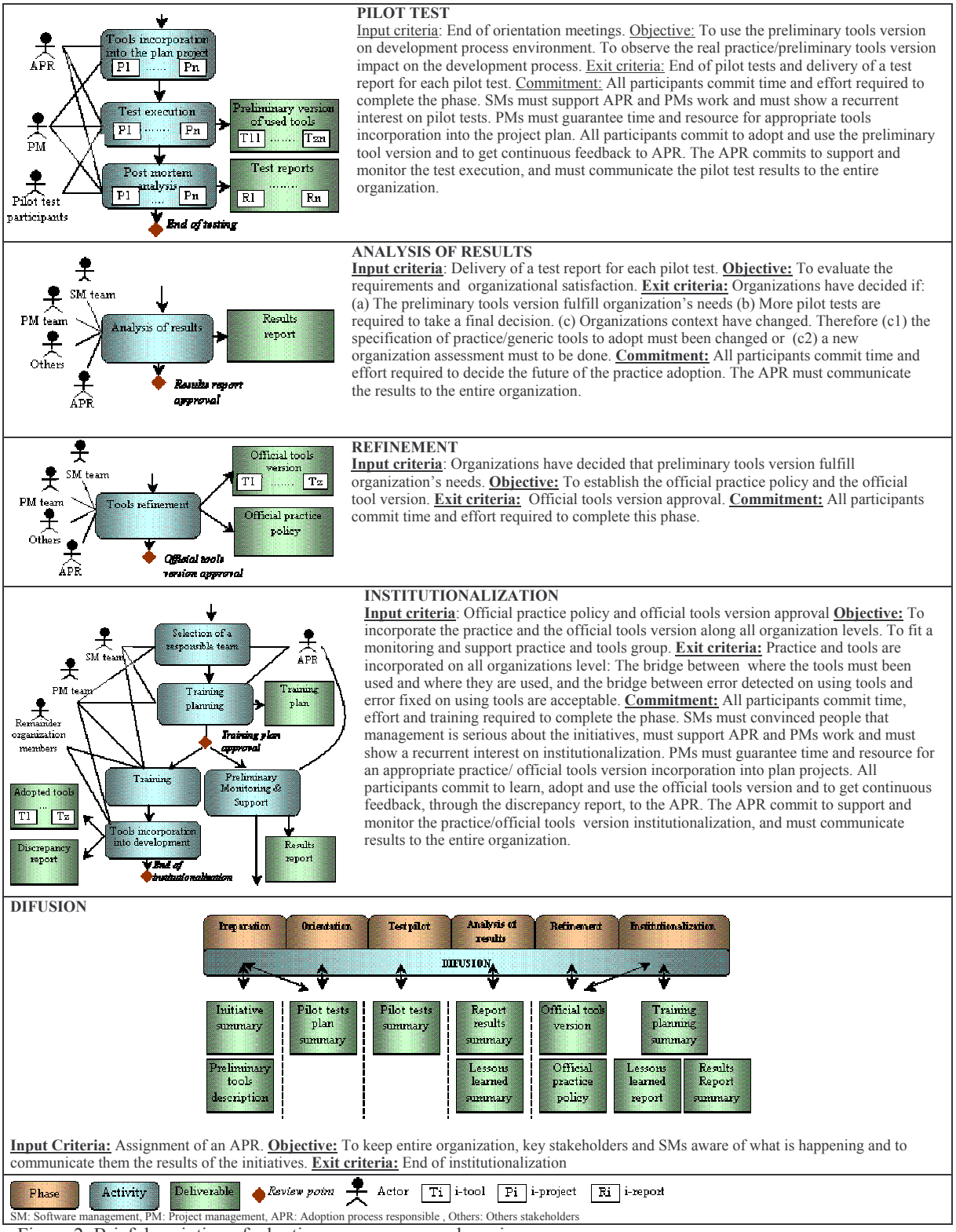


Figure 2. Brief description of adoption process – second version

6.2 Improved quality model, measurement model and model integration

The second version of the quality model was the result of improving attributes considering as keys its effectiveness and efficiency in supporting product and process quantitative management (See Tables 5 thru 8). Note that in the second version the external and internal quality views differ significantly, the number of metrics to be collected has decreased considerably and most of the attributes are now related to time, effort, size and defects because those were identified as the most relevant to manage the quality of implanting generic tools for software practices. Besides, counting rules and measurement units were defined in more precise terms so that models can be more effectively integrated.

Table 5. Internal view of the quality model – second version

Internal View		
Characteristics	**Subcharacteristics**	**Attributes**
Functionality	Fitness	satisfaction
	Usefulness	satisfaction
	Completeness	coverage, missing, excess
Understandability		queries
	Uniformity	discrepancies
Integrity		satisfaction
Operability		queries, errors
Testability		coverage
Cost		effort, time

Table 6. External view of the quality model – second version

External View		
Characteristics	**Subcharacteristics**	**Attributes**
Functionality	Fitness	satisfaction
	Usefulness	satisfaction
	Completeness	missing, excess
Understandability		queries
Operability		queries, errors

Table 7. Example of measurement model – second version

Characteristics	**Subcharacteristics**	**Attributes**	**Counting Rule (Unit)**
Functionality	Fitness	satisfaction	at the end of training assess the participants satisfaction level about tool fitness for purpose using a simple checklist and assigning a simple numeric grade (integer/real)
	Completeness	missing	at the end of training count during an established time period the number of not implanted items from the list of requested and specified items for tool usage (integer)
Operability		queries	at the end of training count during an established time period the number of queries about use and content of tool (integer)
		errors	at the end of training count during an established time period the number of errors regarding use of tool (integer)

Table 8. Example of the models integration – second version

Object Type (Activity): Analysis of the practice and generic tools to adopt			
Characteristic	**Subcharacteristic**	**Attributes**	**Counting rule (unit)**
Cost		effort	invested person hours from start thru specification approval of practice/tools to be implanted (person hours)
		time	invested calendar time from start thru specification approval of practice/tools to be implanted (days)
Object Type (Review point): Specification approval			
Characteristic	**Subcharacteristic**	**Attributes**	**Counting rule (draft)**
Functionality	Fitness	satisfaction	True: done; False: not done (boolean)
Object Type (deliverable): Specification of practice and generic tools to adopt			
Characteristic	**Subcharacteristic**	**Attributes**	**Counting rule (draft)**
Functionality	Fitness	satisfaction	during review of document approval assess using a simple checklist the reviewers degree of satisfaction about the document using a simple grade scale (integer/real)
	Completeness	coverage	(items covered in the document) / (item defined in the document templates) (real)
Understandability		queries	during review of document approval record the number of queries about content (integer)

6.3 Analysis of second version

The adoption process is a very complex process which involved several factors. These are different needs to address in different organizations depending on a number of factors and business context. It is impossible and unrealistic to intend to define an adoption process that can be applied directly on any context. Therefore, the preparation phase of the model includes an adoption process planning activity, which must start adapting the adoption process proposal to the organization context.

However, organizations are rapidly changing and adapting. The adoption model must be flexible enough, so an iterative process was defined in order to re-assess goals and adoption process itself at any point.

During practice adoption many people are involved: senior management, project management and developers (practitioners). They speak different languages and might have different viewpoints on the same situation. However, the adoption model was defined in a way that any person involved can understand the global context of the process and its role and responsibility, the APR must adapt the model to satisfy the organizational context.

As has been mentioned above, organizational environment is rapidly changing. However, many people have a natural aversion to change that manifests as excuses like: “We don’t have time to learn all this new stuff”, “It looks too complicated”, “We heard that this would not work for us because….”. Therefore the human factor must be considered seriously in any SPI effort. Any adoption process must provide effective training, motivate people for using new approaches, methodologies, and technologies, identify clear roles and responsibilities and enable all people to enhance and adapt their skills continuously. In order to cover these aspects, the new version of tool adoption process considers: (1) A diffusion phase to motivate and keep informed the whole organization about initiative objectives and results. This activity focuses on providing visibility to the process. (2) An orientation phase to train and motivate participants on pilot tests. (3) Activities, during the institutionalization, to plan, execute and support the training of people for using the new approach throughout the organization, and (4) the explicit definition of training plans, during orientation and

institutionalization that take into account time and resources as well as an adequate support to allow each individual the implantation of tools as well as improving their own skills.

7 CONCLUSIONS

The adoption process of software practices trough a set of generic tools, such us procedures, templates, checklist and guides, is a complex process that affects the organization's strategies and competitiveness. It must be treated as a serious and relevant project. Therefore, organizations need adoption process frameworks to support this kind of initiative.

In this paper we presented the evolution of an adoption process based on a measurement approach and two cases studies. These experiences have been instrumental to confirm the usefulness and efficiency of this kind of framework. In summary it helps the adoption process monitoring and control and increases its formality and results visibility. All of this translates into a more objective perception and assessment of the adoption process and increase of commitment at all organization levels.

Besides the positive results there are a number of deficiencies in the first version of the proposal. These are mainly the ambiguity of models, the need for explicit planning and project progress diffusion activities. The second version of tool adoption process solves the problems faced when the first version was used, and next we plan to test it in different contexts, develop some tailoring mechanisms, and include people-related issues.

REFERENCES

[1] Bϕegh J., De Panfilis S., Kitchenham B. and Pasquini A. A Method for Software Quality Planning, Control, and Evaluation. IEEE Software, pp. 69-77, March/April 1999.
[2] Garcia S., *"Are you prepared for CMMI?"*, Crosstalk, March 2002, pp. 19-23.
[3] Grady R. Successful Software Process Improvement. Hewlett-Packard Professional Books, 1997.
[4] Kitchenham B., Linkman S., Bϕegh J., Pasquini A. , De Panfilis S., Anders U. SQUID Conceptual Handbook. Technical Report D3.7/1, Esprit Project Nº P8436, February 1996.
[5] McFeely R.. IDEAL: a User´s Guide for Software Process Improvement. Technical Report CMU/SEI-96-HB-001, CMU/SEI, Pittsburgh, Pennsylvania, 1996.
[6] Messnarz R., Colin Tully, *"Better software practice for business benefit: principles and experiences"*, IEEE Computer Society, 1999.
[7] Rojas C., Visconti M., "*Formulación y Aplicación de un Proceso de Análisis de Defectos de Software*". Proceedings III Workshop de Ingeniería de Software, November 2003, Chillán, Chile.
[8] Turner R., *"A study of best practice adoption by defense acquisition programs"*, Crosstalk, May 2002, pp. 4-8.
[9] Visconti M., Guzmán L., *"A measurement-based approach for implanting SQA and SCM practices"*, XX International Conference of the Chilean Computer Science Society, 2000, 126-134.

Geração de Sistemas de Gestão de Conteúdo com Softwares Livres

Fernando Silva Parreiras
Universidade Federal de Minas Gerais, Escola de Ciência da Informação,
Belo Horizonte, Brasil, 31270-010
fparreiras@ufmg.br

Marcello Peixoto Bax
Universidade Federal de Minas Gerais, Escola de Ciência da Informação,
Belo Horizonte, Brasil, 31270-010
bax@ufmg.br

Abstract

This article handles the content management systems generation using open source softwares. It asks if it is possible, based on the main existing theoretical contributions and in the available frameworks, to implement a process that allows reflecting a knowledge domain, represented by models, in a content management system. It is intended to establish the steps necessary for implementation of this approach and to raise framework of content management system development. It was made case study using a software development process as scope. As result, a content management system was generated and a process made up of five activities: (1) platform independent modeling, (2) platform specific modelling, (3) source code generation, (4) source code maintenance and (5) installation. This process uses 50% of the time planed for of the system development, but it presents some restrictions. It concludes that (1) the UML is a good candidate to a knowledge representation language, (2) modelling using levels of abstraction is a crucial necessity for the evolution of the area of information systems, and (3) the used framework presents advantages in prototyping or application in reduced knowledge domains.

Keywords: Domain Modeling, Software Engineering, Knowledge representation, Content Management.

Resumo

Este artigo trata a geração de sistemas de gestão de conteúdos, usando softwares livres. Pergunta-se se é possível, com base nas principais contribuições teóricas existentes e nos arcabouços disponíveis, implementar um processo que permita refletir um domínio de conhecimento, representado por meio de modelos, em um sistema de gestão de conteúdos. Pretende-se estabelecer os passos necessários para implementação desta abordagem e levantar um arcabouço de desenvolvimento de sistemas de gestão de conteúdos. Foi realizado um estudo de caso tendo como escopo o domínio de conhecimento de um processo de desenvolvimento software. Como resultado, tem-se um processo de geração, composto de cinco atividades: (1) Modelagem Independente de Plataforma, (2) Modelagem Específica de Plataforma, (3) Geração do código-fonte, (4) Manutenção Evolutiva do código-fonte, e (5) Instalação. Este processo de geração diminui em mais de 50% o tempo estimado para desenvolvimento do sistema, mas apresenta algumas restrições. Conclui-se que: (1) a UML é uma boa candidata à linguagem de representação do conhecimento, (2) a modelagem em vários níveis é uma necessidade crucial para a evolução da área de sistemas de informação, e (3) o arcabouço utilizado apresenta vantagens na prototipagem ou aplicação em domínios de conhecimento reduzidos.

Palavras-chave: Modelagem de domínio, Engenharia de Software, Representação do conhecimento, Gestão de conteúdo.

1. INTRODUÇÃO

Nos últimos anos, com a rápida depreciação do valor investido no código-fonte de sistemas, pesquisas sobre modelagem em camadas emergem da engenharia de software ([1], [2]), tentando separar em distintos níveis de abstração a representação de um domínio de conhecimento, de forma que o especialista de domínio seja capaz de modelar em nível diferente do analista de sistemas..

Por outro lado, a explosão das comunidades de desenvolvimento de softwares livres, a criação de repositórios de projetos e adoção deste tipo de software pelas organizações e governos fomenta o desenvolvimento de um cenário onde o conhecimento sobre que ferramenta resolve um dado problema é bastante valioso.
Uma terceira questão, mais explorada, é a grande quantidade de conteúdo digital gerado nas organizações. Com a necessidade de compartilhar documentos de maneira rápida e fácil utilizando a web, surgem os sistemas de Gestão de Conteúdo (SGCs). Gestão de Conteúdo é uma abordagem que surge em função da explosão de conteúdo multimídia na web e em Intranets e visa a permitir a gerência de todas as etapas, desde a criação até a publicação de conteúdo [3].
Estas três questões incentivam o desenvolvimento de pesquisas que envolvam a geração de sistemas de gestão de conteúdo baseados de softwares livres. Este artigo tem como problema de pesquisa verificar até que ponto é possível, com base em contribuições teóricas e em arcabouços de ferramentas disponíveis em código aberto, implementar um processo que permita refletir um domínio de conhecimento modelado em um sistema de gestão de conteúdo (SGC).
Com o aumento do número de sistemas de informação utilizados nas organizações, a possibilidade de gerá-los automaticamente deve ser considerada e alguns fatores que contribuem para viabilizar esta geração são:

- A existência de uma linguagem de modelagem como a UML [4], padrão de fato, reconhecido e usado por grande parte da indústria de software.
- A noção de perfis UML que permite a criação de diversos níveis de abstração sucessivos durante a modelagem, separando o conhecimento em diversas camadas ([5], [1], e [2]).
- A existência de uma linguagem-padrão para representação de dados e metadados, a linguagem de marcação estendida XML [6], e o padrão de intercâmbio de metadados, XML Metadata Interchange ou XMI [7].
- Melhor conhecimento sobre padrões e arquiteturas de software de qualidade, como resultado de anos de experiências, reflexão e reengenharia das melhores práticas, em numerosos projetos ([8], [9], [10] e [2]).

Neste trabalho, a Seção 2 apresenta a arquitetura dirigida por modelos, descrevendo os tipos de modelos e suas principais características, a Seção 3 apresenta o processo de geração de sistemas de gestão de conteúdo proposto, sua estrutura, suas atividades e o arcabouço utilizado. A Seção 4 traz um estudo de caso levantando os aspectos positivos e as restrições do processo. Finalmente, conclui-se o artigo com a Seção 5.

2. ARQUITETURA DIRIGIDA POR MODELOS

Em 2001, o Object Management Group (OMG) definiu uma abordagem de modelagem chamada Arquitetura Dirigida por Modelos – Model Driven Architecture (MDA). A abordagem consiste na adoção de modelos em diferentes níveis de abstração, com características distintas, a fim de isolar os aspectos computacionais do domínio de conhecimento.
A questão-chave da arquitetura MDA é a transformação de modelos [2] (Figura 1). O modelo inicial é um modelo independente de plataforma – Platform Independent Model (PIM) –, que possui as seguintes características:

- Representa o domínio de conhecimento sem a distorção de aspectos tecnológicos;
- É completamente independente da pilha de plataforma;
- É um modelo detalhado e, geralmente, representado em UML;
- É a base para o modelo específico de plataforma – Platform Specific Model (PSM).

O modelo PIM é transformado em modelo específico de plataforma – Platform Specific Model (PSM), por intermédio de mapeamentos. Um modelo PSM:

- Contém informações do domínio e da plataforma;
- É criado a partir do mapeamento de um PIM para uma plataforma específica;
- É um modelo detalhado e sempre representado em UML;
- É a base para o código-fonte.

O PSM é, então, convertido em código-fonte e outros artefatos como documentos, roteiros, etc.

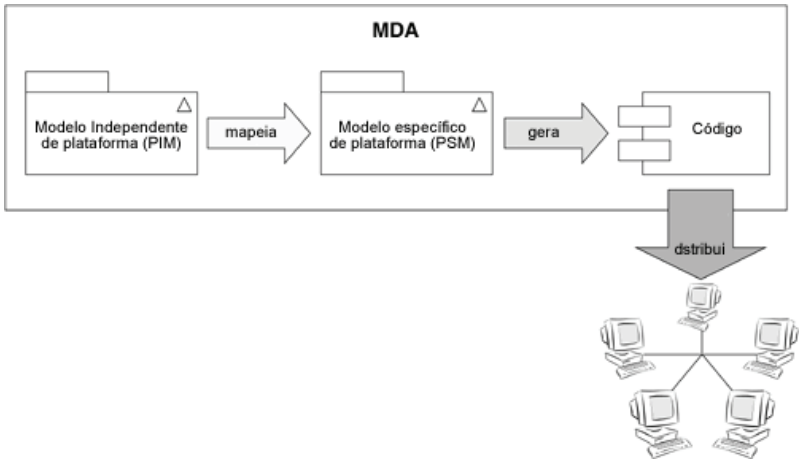


Figura 1. Transformação de modelos [5]

O valor investido em código-fonte tende a depreciar-se rapidamente, juntamente com a linguagem de programação e plataforma ao qual ele pertence. A vantagem desta abordagem é que o domínio de conhecimento fica independente do código-fonte. Na arquitetura MDA, o valor é investido em modelos abstratos. Assim, o valor do modelo aumenta de forma diretamente proporcional ao nível de abstração. Ainda nesta abordagem, o especialista de domínio trabalha em nível de abstração diferente do analista ou programador.
A arquitetura MDA pode trazer contribuições importantes, a saber: permite a separação do conhecimento em níveis de abstração, isolando diferentes aspectos deste; habilita o especialista de domínio a lidar com ferramentas e conceitos mais próximos de sua realidade; permite ao analista ou programador se libertar da responsabilidade do conhecimento de domínio; e abre espaço para o surgimento de ferramentas de geração automática de sistemas, uma vez que, para tanto, basta concentrar esforços no mapeamento entre os diversos níveis de abstração.

2.1 Trabalhos relacionados

Identificaram-se na literatura algumas iniciativas relacionadas à geração de sistemas de informação, apoiadas na arquitetura dirigida por modelos. Sem exaurir o tema, apresentam-se aqui algumas propostas.
Utilizando uma implementação de MDA, [11] mapeiam um MOF (Meta Object Facility) para a linguagem Java. Um MOF [12] define uma linguagem abstrata e uma estrutura para especificação, construção e gerenciamento de metamodelos independentes de tecnologia de implementação. Entretanto, esta implementação não conta com as funcionalidades pré-construídas de um ambiente de Gestão de Conteúdo e demanda grande quantidade de esforço na implementação de serviços como personalização, segurança, etc.
Outro projeto que se baseia na arquitetura MDA é o projeto "XIS" [13]. "XIS" (Desenvolvimento de Sistemas de Informação baseado em Modelos e Arquiteturas de Software) é um projeto de Investigação e desenvolvimento realizado no contexto do Grupo de Sistemas de Informação do LAVC/INESC-ID[1]. O principal objetivo deste projeto é o estudo, desenvolvimento e avaliação de mecanismos de produção de sistemas de informação de forma mais eficiente. O projeto XIS é influenciado pelo modelo de referência MDA e baseia-se fortemente em um conjunto de práticas emergentes, seguindo uma abordagem (1) baseada em modelos, (2) centrada em arquiteturas de software, e (3) baseada em técnicas de geração automática. Contudo, este projeto também não usufrui as características já presentes nos ambientes de Gestão de Conteúdo, e nem da facilidade de incorporação de novas funcionalidades.
[14] descrevem o Modelo de Objeto Adaptável (Adaptative Object-Model) tratado também como arquitetura reflexiva ou meta-arquitetura. Um Modelo de Objeto Adaptável (MOA) é um sistema que representa classes, atributos e relacionamentos como metadados. Ele é um modelo baseado em instâncias, ao invés de classes. Usuários mudam o metadado (modelo de objeto) para refletir alterações no domínio. Estas alterações modificam o comportamento do sistema. MOA é uma arquitetura que pode, dinamicamente, adaptar-se, em tempo de execução, aos novos requisitos do usuário. Outras nomenclaturas para esta arquitetura são "modelo de objeto ativo" [15] e "modelo de objeto dinâmico" [16]. Esta abordagem concentra-se no aspecto evolutivo do domínio de conhecimento. Em contrapartida, sistemas de informação que utilizam o MOA são mais difíceis de construir e de serem compreendidas pelos atores envolvidos, além de não especificar como se dá a interface com o usuário.

[1] http://berlin.inesc-id.pt

Uma outra implementação em camadas é realizada pelo projeto Hércules [17]. O projeto Hércules consiste em um arcabouço para desenvolvimento de sistemas de informação, visando possibilitar a geração automática de código a partir de diagramas UML, e estabelece um conjunto de descrições de alto nível de abstração, que os analistas ou programadores utilizam para especificar a apresentação, o desenvolvimento operacional e a persistência do sistema a ser construído. Este modelo é dividido em três camadas: domínio controle e apresentação. Cada uma destas camadas possui diagramas UML associados. Embora esta abordagem envolva a geração automática de sistemas, ela não se preocupa em tratar separadamente o conhecimento de domínio do conhecimento tecnológico..

3. PROCESSO DE GERAÇÃO DE SISTEMAS DE GESTÃO DE CONTEÚDO

O Processo de geração proposto é composto de um arcabouço tecnológico, baseado em ferramentas de código aberto, e um conjunto de atividades, que produzem e consomem resultados. O arcabouço e as atividades são detalhados a seguir.

3.1 Arcabouço

O arcabouço utilizado é baseado em softwares de código aberto. A partir de uma busca no sítio web sourceforge[2], identificou-se sistemas que apoiassem às atividades necessárias à realização do processo. A partir de descritores como "Content Management System" e "UML", foram encontradas ferramentas que atendem a fins de Gestão de Conteúdo e edição em UML. Foram escolhidas ferramentas que possuíam a maior comunidade de desenvolvedores e se encontravam no estágio de maturidade estável (5 - stable).

O arcabouço utilizado no experimento é formado pelo conjunto de ferramentas listadas abaixo e descritas na Tabela 1.

- Editor UML: Gentleware Poseidon UML (GENTLEWARE);
- Servidor de aplicações, de banco de dados e web: Zope [18];
- Plataforma de Gestão de Conteúdo (SGC): Plone [19];
- Componentes pré-construídos: Plone Archetypes [20];
- Gerador de código-Fonte: ArchGenXML [21].

Tabela 1. Arcabouço Utilizado

Nome da ferramenta:	Tipo de licença	Versão	Descrição da função	No. de envolvidos
Poseidon UML CE	Livre para uso não comercial.	3.0.1	Editor UML. Utilizado para modelagem OO na notação UML.	21
Plone	GPL	2.0.4	Sistema de Gestão de Conteúdos.	88
Archetypes	GPL	1.3.1.	Ferramenta de geração automática de tipos de conteúdo plone, orientada a esquemas.	65
ArchGenXML	GPL	1.1	Utilitário de linha de comando que gera aplicações Plone baseado no framework Archetypes, a partir de um modelo UML representado em XMI ou XMLSchema.	65
strip-o-gram	GPL	1.4	Converte a documentação HTML disponível no arquivo xmi para texto puro.	6
oi18ndude	GPL	0.2.2	Permite a geração de rótulos traduzíveis.	81
Zope	GPL	2.7	Servidor de aplicações, banco de dados e web.	-

A interação entre estas ferramentas e os atores envolvidos pode ser ilustrada na Figura 2. O especialista de domínio gera o Modelo Independente de Plataforma (PIM), que é mapeado em um Modelo Específico de Plataforma (PSM) com a ajuda de um analista ou programador. O PSM é o insumo do gerador de código-fonte (ArchGenXML), produzindo o código-fonte (CF). Este, juntamente com componentes pré-construídos são os insumos da plataforma de gestão de conteúdo que produz o sistema de gestão de conteúdo (SGC) para um domínio de conhecimento específico.

[2] http://www.sourceforge.net

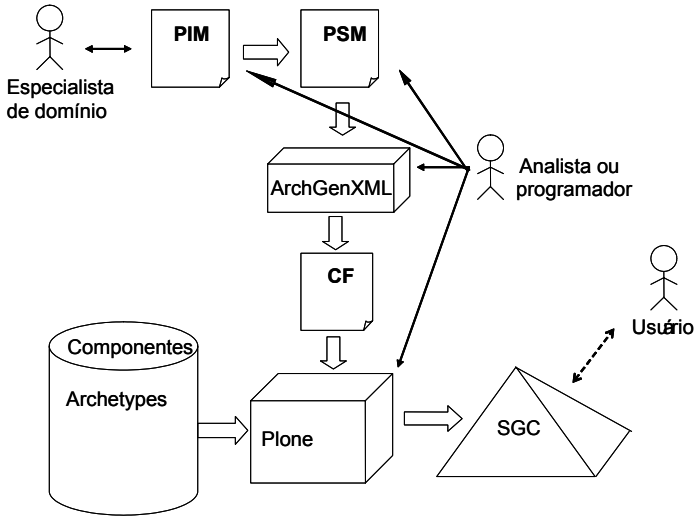


Figura 2. Arcabouço utilizado

3.2 Estrutura do Processo

O processo utiliza um arcabouço composto de ferramentas, que possuem como requisito outras ferramentas. O processo é composto de atividades, que assumem compromissos. Estas atividades produzem resultados e consomem resultados produzidos nas atividades anteriores (Figura 3).

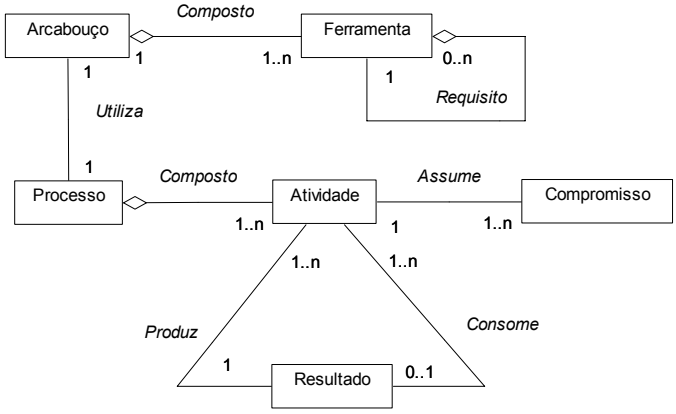


Figura 3. Estrutura do processo

3.3 Atividades do processo

As cinco atividades do processo são: Modelagem Independente de Plataforma; Modelagem Específica de Plataforma; Geração do código-fonte; Manutenção Evolutiva do código-fonte; e Instalação (Tabela 2). O fluxo das atividades é ilustrado na Figura 4, onde é são apresentados os insumos e produtos de cada atividade, assim como o ator relacionado.

Tabela 2. Atividades e resultados

Resultado	Sigla	Responsável	Ferramenta aplicável	Atividade
Modelo Independente de Plataforma	PIM	Especialista de Domínio	Editor UML	Modelagem Independente de plataforma
Modelo Específico de Plataforma	PSM	Analista ou Programador	Editor UML	Modelagem Específica de Plataforma

Código-fonte	CF	Analista ou Programador	Gerador de código	Geração do código-fonte
Código-fonte Evoluído	CF	Analista ou Programador	Editor de Código	Manutenção Evolutiva do código-fonte
Sistema de Gestão de Conteúdo	SGC	Analista ou Programador	Plataforma de gestão de conteúdo	Instalação

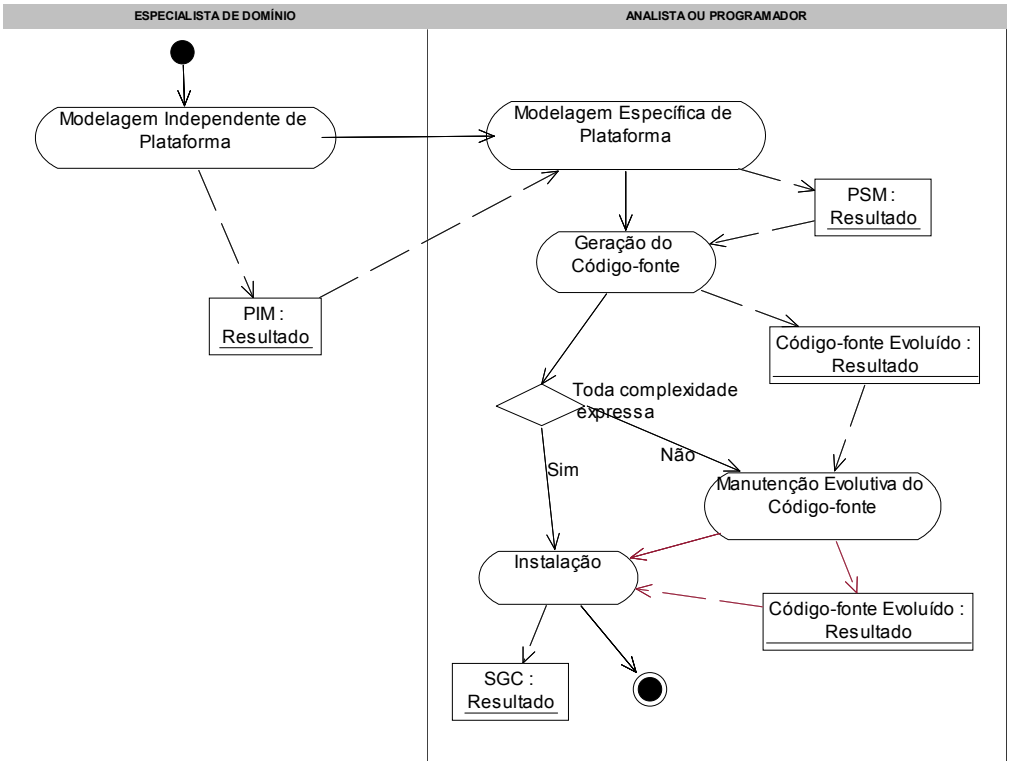


Figura 4. Atividades do Processo

Outra forma de sistematizar as transformações ocorridas neste experimento é por intermédio da equação abaixo:

SGC = Plone (MAN (CF (PSM (PIM, Perfil_UML), ArchGenXML)))

Em que:

- SGC: Sistema de Gestão de Conteúdo.
- Plone: Plataforma de gestão de conteúdo que utiliza o PraxisCMF;
- MAN = Manutenção Evolutiva do código-fonte do SGC;
- CF = Transformação do PSM, a partir da ferramenta ArchGenXML, em Código-fonte;
- PSM = Transformação do PIM e Perfil_UML em Modelo Específico de Plataforma;
- PIM: Modelo independente de plataforma;
- Perfil_UML: conjunto de mecanismos de extensão;

Abaixo são detalhadas as atividades do processo.

3.3.1 Modelagem independente de plataforma
Esta atividade concentra esforços na modelagem e representação de um determinado domínio de conhecimento. Ao modelar um domínio de conhecimento, Noy e McGuinness (2001) recomendam, inicialmente, a identificação de modelos já existentes. Utilizou-se o editor UML Poseidon para a criação do diagrama de classes. Esse editor utiliza o formato XMI [7] para armazenamento do modelo. O formato XMI é utilizado na geração automática também em [13] e [17]. A arquitetura MDA [2] também recorre ao formato XMI. O resultado desta atividade é o Modelo Independente de Plataforma (PIM), representado em UML e armazenado no formato XMI.

3.3.2 Modelagem específica de plataforma
Os aspectos dependentes de plataforma são adicionados nesta atividade, que cria um Modelo Específico de Plataforma (PSM) com menor nível de abstração, mais próximo do código-fonte. A partir de um perfil UML, é possível acrescentar aspectos inerentes a um determinado arcabouço. Utilizou-se o perfil UML ArchGenXML [21], que permite a representação de ícones, rótulos de dados, máscara de entrada, etc. A adequação tecnológica ocorre nesta atividade, em que o analista ou programador modela de acordo com a tecnologia escolhida.
Assim, é necessário manter um sincronismo entre o PIM e o PSM. Propõe-se um mapeamento entre os modelos, por meio de um roteiro de transformação. Este roteiro é escrito na linguagem XML Extensible Stylesheet Language Transformation (XSLT) [22], manualmente, pelo analista ou programador, e contem todas as transformações realizadas. Um mapeamento deste tipo também é utilizado por Silva [13], e outros mapeamentos semelhantes são realizados em ontologias, conforme [23].
O resultado desta atividade é o PSM, representado em UML, estendido segundo um perfil UML e armazenado em XMI.

3.3.3 Geração do código-fonte
Nesta atividade, o PSM é transformado em código-fonte, por intermédio de um gerador de código-fonte em linguagem Python, que importa componentes pré-construídos disponíveis no arcabouço. Este código-fonte compõe o SGC. O PSM serve de entrada para o gerador de código-fonte, responsável por transformar o PSM em CF. Esta atividade resulta no código-fonte (CF) do SGC.

3.3.4 Manutenção Evolutiva do código-fonte
Se toda a complexidade do SGC puder ser expressa no PSM, esta atividade é opcional. Contudo, dado o atual desenvolvimento das ferramentas do arcabouço utilizado, esta ocorrência é rara. Assim, o analista ou programador deve fazer alterações no CF, a fim de atender os requisitos do sistema de informação que não puderam ser expressos no PIM ou PSM. O CF é, então, copiado e evoluído, tornando-se o resultado desta atividade, o Código-fonte Evoluído (CFE).
Esta alteração traz a mesma necessidade de sincronismo que a atividade de Modelagem Específica de Plataforma. Entretanto, como o CF é estruturado, mas não em uma linguagem de marcas, o mapeamento por meio de um roteiro de transformação fica mais difícil. Embora mais difícil, tal mapeamento é possível, utilizando técnicas de engenharia reversa, que consiste na leitura do CF do SGC para geração de modelos mais abstratos. No processo, se ocorre uma alteração em uma das atividades anteriores, e existe a necessidade de manutenção do código-fonte, esta alteração deve ser tratada manualmente nesta atividade. Esta atividade resulta no CFE.

3.3.5 Instalação
Nesta atividade, o código-fonte manipulado na atividade anterior é agrupado, formando o SGC. O roteiro de instalação envolve a interrupção do serviço web, cópia do código-fonte e outros arquivos, inicialização do serviço web e instalação do SGC na plataforma de gestão de conteúdo.

4. ESTUDO DE CASO

4.1 Escopo
Foi realizado um estudo de caso com o domínio de conhecimento do processo de desenvolvimento de software que, de acordo com o aumento do grau de maturidade, sofre alterações freqüentes. A motivação de uso deste domínio para o estudo de caso foi principalmente seu dinamismo. Das 30 empresas, hoje, no Brasil, que possuem algum nível de certificação CMM [24], verifica-se que 24 delas estão no nível 2 e cinco no nível 3. Há uma empresa no nível 4 e nenhuma no nível 5. [25] constata que as empresas nacionais buscam a melhoria de seu processo de software, alterando-o e adequando-o às melhores práticas internacionais. Com efeito, para controlar os indicadores necessários para os níveis 2 e 3 do CMM é indispensável o uso de sistemas de apoio. Elegeu-se um processo de desenvolvimento de software baseado no Processo Unificado [26] chamado Praxis [27].
Apenas uma parte do domínio foi utilizada. Foram considerados apenas os diagramas de classe, que contemplam itens estruturais do domínio. Esta restrição se deu devido a uma das ferramentas do arcabouço utilizado, o gerador de

código-fonte, que manipula somente diagramas de classe. Foram identificadas 45 classes e 49 relacionamentos em todos os diagramas de classe do Praxis. O experimento utilizou um modelo com 19 classes, 26 relacionamentos e 38 atributos.
A medição do tamanho do SGC deste domínio resulta em 247 pontos de função não ajustados. Ajustando este valor baseado no ambiente tecnológico obtém-se 207,48 pontos de função ajustados (PFA). Aplicando um coeficiente de produtividade razoavelmente baixo neste valor, ou seja, 1 (um), tem-se uma estimativa de 207,48 horas a serem gastas no desenvolvimento do SGC.

4.2 Resultados

Neste estudo de caso foram gastas, aproximadamente, 66 horas para o desenvolvimento do SGC, conforme descrito na Tabela 3. Sendo constantes todas as outras variáveis, observa-se uma diferença significante entre os valores obtidos pelo autor e os valores estimados conforme a métrica de análise por pontos de função, salvo as possíveis falhas na metodologia utilizada. Esta diferença pode ser atribuída principalmente ao gerador de código-fonte e aos componentes pré-construídos. O Gerador de código-fonte permite gerar automaticamente inferfaces de inclusão, alteração e exclusão dos objetos, enquanto que os componentes pré-construídos trazem funcionalidades como busca, navegação, padrões de apresentação e outros.

Tabela 3. Tempo gasto pelo autor para implementação

Atividade	Tempo (em horas)
Modelagem independente de plataforma	24
Modelagem específica de plataforma	24
Geração do código-fonte	1
Manutenção Evolutiva do código-fonte	16
Instalação	1
TOTAL	66

Os valores utilizados para o cálculo do esforço de desenvolvimento são estimados, devido à ausência de séries históricas de projetos desenvolvidos utilizando a mesma plataforma tecnológica, o que apresenta uma fragilidade na comparação entre o esforço estimado e o esforço realizado.
Além da utilização do processo para a geração do sistema de gesta de conteúdo, realize um papel importante no desenvolvimento de software: a prototipagem. A prototipagem consiste na elaboração de um protótipo, ou seja, versão parcial e preliminar do sistema destinada à validação com o usuário ou teste. Após a atividade de modelagem, comum a qualquer processo de desenvolvimento de sistemas de informação, uma solução interessante seria utilizar esta implementação para gerar uma versão preliminar do sistema para apreciação do usuário, o que reduziria ainda mais o tempo de geração, pois o tempo gasto na manutenção do código-fonte seria menor.

4.2.1 Principais Restrições

As ferramentas de código aberto disponíveis ainda não suportam todo o poder da modelagem orientada a objeto. A impossibilidade de usufruir todos os recursos da linguagem UML restringiu o potencial de representação utilizado.
Um exemplo destas restrições é a utilização de apenas um dentre os nove diagramas da linguagem UML pela plataforma adotada no estudo de caso, o diagrama de classes. Diagramas com o poder de expressão comportamental como o diagrama de atividades, diagrama de estado e diagrama de seqüência ainda não são considerados pelas ferramentas disponíveis, o que restringe bastante o escopo de aplicação. Por exemplo, a consideração de "fluxos de trabalho" modelados ainda não pode ser integrada ao processo de geração.
A usabilidade do SGC gerado apresenta algumas restrições como, por exemplo, no que diz respeito às referências cruzadas. Ao acessar um determinado objeto, tem-se uma visualização dos outros que se relacionam com ele. Entretanto, não se pode visualizar a descrição do relacionamento, o que se torna um agravante quando um objeto se relaciona com outro de duas formas diferentes.
O mapeamento entre as camadas também é um ponto que apresenta restrições. Enquanto o mapeamento entre PIM e PSM deve ser escrito manualmente pelo analista ou programador, o mapeamento entre o PSM e os CFs fica mais complicado, pois embora os CFs sejam estruturados, ainda não há uma forma automática de realizar este mapeamento.

5. CONCLUSÃO

Este artigo apresentou um processo de geração de Sistemas de Gestão de Conteúdo a partir do uso de softwares livres. Apresentou, ainda, um estudo de caso descrevendo os aspectos positivos e restrições na aplicação do processo. Conclui-se que:

- A modelagem em vários níveis é uma necessidade crucial para a evolução da área de sistemas de informação, principalmente quando considerados domínios em que o número de conceitos é muito grande e suas propriedades e

relações muito dinâmicas, como no caso de aplicações no domínio da medicina ou mesmo da engenharia de software. Nesses domínios essa separação parece fundamental para a perenidade dos sistemas de informação;

- O arcabouço utilizado apresenta vantagens, aplicado na prototipagem ou geração de SGCs;
- Quando um sistema utiliza o paradigma da orientação a objeto, a UML é uma boa candidata à linguagem de representação do conhecimento.
- O uso de um gerador de código-fonte e de objetos pré-construídos para a geração de um SGC reduz consideravelmente o tempo de desenvolvimento do mesmo.
- A modelagem em camadas, utilizando diagramas e perfis UML, apresenta uma forma interessante de adicionar aspectos da implementação no modelo de forma isolada, sem comprometer a visão do usuário. Contribui, ainda, para a documentação do projeto, que nem sempre é concebida dentro do processo de software, ou é considerada uma atividade sacrificante por parte dos responsáveis.

Propõe-se que este trabalho continue evoluindo nas seguintes direções:
- Continuação do desenvolvimento do arcabouço utilizado, já que as ferramentas que o constituem não param de evoluir em seus princípios teóricos;
- Aplicação do arcabouço utilizado e do processo de geração utilizado no desenvolvimento de sistemas de gestão de conteúdos em outros domínios, buscando a compreensão do problema a partir da sua consideração sob outras perspectivas;

Referências

[1] Beale, T. Archetypes: Constraint-based domain models for future-proof information systems. In: OOPSLA, 17, 2002, Seatlle. Anais eletrônicos... Workshop on behavioural semantics, Seatlle: ACM. 2002.

[2] Frankel, D. S. Model Driven Architecture: Applying MDA to Enterprise Computing. Indianapolis: Wiley Publishing, Inc., 2003. 352p.

[3] Pereira, J.C.L., Bax, M. P. Introdução à Gestão de Conteúdos. In: KM BRASIL, 2002, São Paulo. Anais (CD-ROM)... São Paulo: [s.n.], 2002.

[4] Booch, G., Rumbaugh J., Jacobson I. UML: guia do usuário. Rio de Janeiro: Campus, 2000. 472 p.

[5] Arlow, J., Neustadt, I. Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML. Boston: Addison Wesley, 2003. 528p.

[6] Yergeau, F. et al. Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation, February 2004.

[7] HEATON, L. XML Metadata Interchange (XMI) Specification. Versão 1.2. Needham: OMG, jan. 2002b. 268 p. Technical report.

[8] Gamma, E. et al. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison Wesley, 1995. 395p.

[9] Buschmann, F. et al. Pattern-Oriented Software Architecture: A system of patterns. 1. ed. Mississauga: John Wiley & Sons, 1996. 476 p.

[10] Hofmeister, C., Nord, R., Soni, D. Applied Software Architecture. Boston: Addison Wesley, 1999. 397p.

[11] Santos, H. L., Barros, R. S. M. Utilizando o MOF na construção de metamodelos em um ambiente MDA. In: Simpósio Brasileiro De Engenharia De Software, 18, 2004, Brasília. Anais... Brasília: Sociedade Brasileira de Computação (SBC), 2004. CD-ROM.

[12] Heaton, L. Meta Object Facility (MOF) Specification. Versão 1.4. Needham: OMG, apr. 2002a. 358 p. Technical report. Disponível em <http://www.omg.org/docs/formal/02-04-03.pdf>. Acesso em 25 jan. 2005.

[13] Silva, A. R. Abordagem XIS ao Desenvolvimento de Sistemas de Informação.In: Conferência da Associação Portuguesa de Sistemas de Informação, 4, 2003, Porto. Anais... Porto: Universidade Portucalense. 2003a.

[14] Yoder, J. W., Balaguer, F., Johnson, R. Architecture and Design of Adaptive Object Models. In: Intriguing

Technology Paper OOPSLA, 2001, Tampa Bay. Anais eletrônicos... Tampa Bay: ACM SIGPLAN Notices, ACM Press, Dez. 2001.

[15] Foote, B, Yoder, J W. Metadata and Active Object-Models. In: Collected papers from the PLoP '98 and EuroPLoP '98 Conference, 1998, Washington. Proceedings... Washington: Washington University. 1998.

[16] Riehle, D., Tilman, M., Johnson, R. Dynamic Object Model. In: PLoP2000, 7, 2000, Monticello. Proceedings... Monticello: Technical Report #wucs- 00-29, Dept. of Computer Science, Washington University, 2000. p. 1-13.

[17] Pais, A. P. V., Oliveira, C. E. T., Leite, P. H. P. M. Robustness Diagram: A Bridge Between Business Modeling And System Design . In: Proceedings of VII International Conference on Object-Oriented Information Systems - OOIS'01. Calgary, Canadá: Springer-Verlag. 2001, v. 1, p. 530-539.

[18] Latteier, A., Pelletier, M. The Zope Book. Berkeley: Pearson Education, 2001. 384p.

[19] Mckay, A. The Definitive Guide to Plone. Berkeley: Apress, 2004. 584p.

[20] Silva, S. Archetypes: An Introdution. In: ZopeMag.com. Kurfürstendamm: beehive KG. 2003b. Disponível em: <http://www.zopemag.com/Issue006/Section_Articles/article_IntroToArchteypes.html>.

[21] Klein, J. W. ArchGenXML Manual - generating Archetypes using UML. 2004. Disponível em: <http://plone.org/documentation/archetypes/archgenxml-manual>

[22] Kay, M. XSLT referência do programador. 2. ed. Rio de Janeiro: Alta Books, 2002. 667p.

[23] Handschuh, S., Staab, S., Volz, R. On deep annotation. In: International World Wide Web Conference (WWW), 12, 2003, Budapest. Proceedings… Budapest: ACM Press, 2003.

[24] Paulk, M. C. et al. Capability Maturity Model for Software, Version 1.1. Pittsburgh: Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, Feb. 1993. 82 p. Technical report.

[25] Weber, K. et al. Modelo de Referência para Melhoria de Processo de Software: uma abordagem brasileira. In: XXX Conferência Latino-Americana De Informática (CLEI2004), 30, 2004, Arequipa. Anais...Arequipa: [s.n], 2004.

[26] Jacobson, I., Rumbaugh, J., Booch, G. Unified Software Development Process. Reading - MA: Addison-Wesley, 1999.

[27] Paula Filho, Wilson de Pádua. Engenharia de software: fundamentos, métodos e padrões. 2. ed. Rio de Janeiro: LTC, 2003. 602p.

Una Experiencia con Estudiantes para la Estimación del Esfuerzo de cada Iteración en Proyectos de Software

José Antonio Pow-Sang Portillo
Pontificia Universidad Católica del Perú, Dpto. de Ingeniería,
Lima, Perú
jpow@pucp.edu.pe

Abstract

Effort and cost estimation is still one of the hardest tasks in software project management. At the moment, there are many techniques to do this job, like Function Points, Use Case Points and COCOMO, but there is not much information about how to use those techniques in non-waterfall software lifecycles such as iterative or spiral lifecycles projects.
This paper shows the results obtained when applying a technique to estimate the effort of each construction iteration in software development projects that use iterative-incremental lifecycles. The results were obtained from software projects of a fourth-year course in Informatics. The students calculated the effort of the second and third iteration of construction on the basis of the results obtained in previous iterations. The technique proposes the use of Function Points and COCOMO II and it has been proved in a previous semester.

Keywords: Effort Estimation, iterative-incremental lifecycle, software engineering experimentation, use cases.

Resumen

La estimación del esfuerzo y costo es una de las tareas más difíciles en la gestión de proyectos de software. En la actualidad, existen técnicas para realizar esta tarea. Algunas de ellas son: Puntos de Función, Puntos de Casos de Uso y COCOMO II. Lamentablemente, no hay mucha información de cómo utilizar las técnicas en mención para ciclos de vida diferentes al de cascada, como los ciclos de vida iterativo-incrementales y en espiral.
Este artículo muestra los resultados obtenidos en la aplicación de una técnica para estimar el esfuerzo de proyectos que usan los ciclos de vida iterativo-incrementales. Estos resultados fueron obtenidos en proyectos realizados por alumnos de cuarto año de la carrera de Ing. Informática. Para esta actividad, los estudiantes calcularon el esfuerzo de la segunda y tercera iteración de la fase de construcción tomando como base los resultados que obtuvieron en iteraciones previas. La técnica propone el uso de Puntos de Función y COCOMO II y ha sido probada en un semestre anterior al de esta experimentación.

Palabras claves: Estimación del esfuerzo, ciclo de vida iterativo-incremental, experimentación en Ing. de Software, casos de uso.

1. INTRODUCCIÓN

La creciente complejidad de los desarrollos software que provocó la denominada "crisis del software" se ha tratado de abordar mediante el planteamiento de nuevos métodos, metodologías, técnicas y paradigmas para minimizar su impacto. El alcance de dichas propuestas no se limita exclusivamente a actividades relacionadas con el desarrollo en sí de los sistemas, sino que abarca también las actividades de gestión de los mismos. Una de estas actividades es la de la estimación de los proyectos software.
A pesar de que la estimación de proyectos continúa siendo una tarea muy compleja, en muchas ocasiones dejada al albur de la pericia del experto estimador, en las últimas décadas se han desarrollado algunas técnicas para la estimación del esfuerzo de proyectos software completos, tales como Puntos de Función [23], Puntos de Caso de Uso [10] y COCOMO II [4]. Aún así, estas técnicas –si bien se postulan como independientes de la tecnología final de desarrollo– fueron concebidas para su aplicación en sistemas basados en el paradigma estructurado con un ciclo de vida clásico o en cascada de Royce [18], y aún es difícil emplearlas en desarrollos orientados a objetos y ciclos de vida iterativo-incrementales, tan en boga en los últimos años. Incluso, parece interesante que éstas técnicas de estimación exploten

para sus propósitos la información proporcionada por prácticas muy extendidas últimamente, como, por ejemplo, la de los casos de uso.
Conociendo las ventajas de realizar experimentos con alumnos [5] y teniendo en cuenta la problemática planteada, el presente artículo muestra una experiencia en la aplicación de una técnica basada en Puntos de Función (PF) que propone un cálculo de estimación del esfuerzo de cada una de las iteraciones. Esta experiencia fue realizada para dos equipos de trabajo conformados por alumnos de 4to año de la carrera de Ing. Informática, en el semestre 2004-2 (agosto-diciembre 2004), dentro de un curso del área de Ing. de Software.
Este documento se ha estructurado de la siguiente manera: la sección 2 muestra un breve resumen de las técnicas de Puntos de Función, COCOMO II y su relación con los casos de uso; la sección 3, la técnica para estimar y planificar la construcción de software para ciclos de vida iterativos-incrementales; la sección 4, la información correspondiente a la experimentación; la sección 5, los resultados obtenidos; la sección 6, una discusión de los resultados obtenidos con ambas técnicas; y, finalmente, se presentan las conclusiones y trabajo futuro de esta experiencia..

2. LAS TÉCNICAS DE ESTIMACIÓN Y LOS CASOS DE USO

Esta sección muestra un breve resumen de las técnicas de Puntos de Función y COCOMO II, y su relación con los casos de uso. Además, se da una visión general del trabajo que se ha encontrado en relación al tema.

2.1 La Técnica de Puntos de Función.

Los Puntos de Función [20] fueron introducidos por Albrecht [1] y su propósito es medir el software cualificando la funcionalidad que proporciona externamente, basándose en el diseño lógico del sistema. Los objetivos de los Puntos de Función son:

- Medir lo que el usuario pide y lo que el usuario recibe.
- Medir independientemente de la tecnología utilizada en la implantación del sistema.
- Proporcionar una métrica de tamaño que dé soporte al análisis de la calidad y la productividad.
- Proporcionar un medio para la estimación del software.
- Proporcionar un factor de normalización para la comparación de distintos software.

El análisis de los Puntos de Función se desarrolla considerando cinco parámetros básicos externos del Sistema: External Input (EI), External Output (EO), External Query (EQ), Internal Logic File (ILF) y External Interface File (EIF).
Con estos parámetros, se determinan los puntos de función sin ajustar (PFsA). A este valor, se le aplica un Factor de Ajuste obtenido en base a unas valoraciones subjetivas sobre la aplicación y su entorno; es decir, las características generales del sistema.

2.2 COCOMO II y los Puntos de Función

COCOMO II, propuesto y desarrollado por Barry Boehm [4], es uno de los modelos de estimación de costos mejor documentados y utilizados. El modelo permite determinar el esfuerzo y tiempo que se requiere en un proyecto de software a partir de una medida del tamaño del mismo expresada en el número de líneas de código que se estimen generar para la creación del producto software.
Debido a la complejidad de los proyectos de software, el modelo original fue modificado, denominándose al modelo actual COCOMO II. El nuevo modelo permite estimar el esfuerzo y tiempo de un proyecto de software en dos etapas diferentes: diseño temprano y post-arquitectura. Para ambas etapas, el modelo propone usar un conjunto de drivers de coste, los cuales indican el contexto en el cual se está desenvolviendo el proyecto. Estos drivers son utilizados para determinar el factor de ajuste "EAF", el cual se utiliza para realizar el cálculo del esfuerzo necesario para completar el proyecto. La tabla 1 muestra la relación de los drivers correspondientes al modelo de post-arquitectura.

Tabla 1: Drivers de coste para el modelo de Post-Arquitectura de COCOMO II

Driver de Coste	Descripción
RELY	Fiabilidad requerida del software
DATA	Tamaño de la base de datos
CPLX	Complejidad del producto
RUSE	Reusabilidad requerida
DOCU	Documentación de acuerdo a las necesidades del ciclo de vida
TIME	Restricción de tiempo de restricción
STOR	Restricción de almacenamiento principal
PVOL	Volatilidad de la plataforma
ACAP	Capacidad de analistas
PCAP	Capacidad de programadores

PCON	Continuidad del personal
AEXP	Experiencia en aplicaciones
PEXP	Experiencia de plataforma
LTEX	Experiencia de lenguajes y herramientas
TOOL	Uso de herramientas de software
SITE	Desarrollo en múltiples lugares

Otro cambio realizado en el nuevo modelo consiste en poder determinar el esfuerzo y tiempo de un proyecto de software a partir de los puntos de función sin ajustar (PFsA), lo cual supone una gran ventaja, dado que en la mayoría de los casos es difícil determinar el número de líneas de código de que constará un nuevo desarrollo, en especial cuando se tiene poca o ninguna experiencia previa en proyectos de software. Esto hace que ambos modelos –Puntos de Función y COCOMO II– sean perfectamente compatibles y complementarios.

2.3. Los Casos de Uso y los Puntos de Función.

Los casos de uso fueron introducidos en 1987 como una herramienta de la técnica Objetory [19]. Su utilización en los procesos de Ingeniería de Software fue propuesta por Ivar Jacobson y publicada en su libro "Object Oriented Software Engineering" [9]. Actualmente, su empleo se ha extendido aún más, debido a su inclusión en UML [12], por lo que su uso en el desarrollo de software orientado a objetos se ha vuelto altamente recomendable, si no obligatorio.

David Longstreet, en uno de sus artículos [11], señala que el análisis de puntos de función se puede aplicar de manera sencilla con los casos de uso mejorando la calidad de los documentos de requerimientos y, a la vez, mejorando la estimación del proyecto de software. El aplicar la técnica de Puntos de Función permite verificar y validar el contenido de un documento de Especificación de Requisitos de Software.

Lo interesante de la aplicación de ambas técnicas es que se puede actualizar el conteo de los puntos de función cada vez que los casos de uso cambien y, de esta manera, determinar el impacto que un caso de uso específico puede producir en la estimación del desarrollo de todo el proyecto.

Thomas Fetcke [7] propone una forma para utilizar la técnica de puntos de función con la metodología OOSE de Jacobson [9]. La relación que muestra se basa en los casos de uso y en el diagrama de clases de análisis propuestos por dicha metodología. En su propuesta no especifica el tipo de ciclo de vida que se debe seguir con su técnica, por lo que se podría inferir que se debería usar el modelo ciclo de vida en cascada.

3. TÉCNICA PARA PLANIFICAR Y ESTIMAR LAS ITERACIONES DE UN PROYECTO DE SOFTWARE CON PUNTOS DE FUNCIÓN

En [13] se propuso una técnica para estimar y planificar las iteraciones en base a puntos de función y COCOMO II y es la que se utilizó para esta experimentación.

La técnica en mención propone dos pasos principales: en primer lugar, determinar los casos de uso a realizar por iteración y, posteriormente, estimar el esfuerzo para cada iteración. A continuación se detallan las actividades que se siguen en cada uno de los pasos mencionados.

3.1 Determinar los Casos de Uso a Realizar por Iteración (paso 1).

El objetivo de esta tarea es determinar los casos de uso que se deberán implementar en cada iteración. Para ello, se deberá haber realizado previamente la especificación de todos los casos de uso del software a construir y que deberán estar en el documento de especificación de requisitos de software (para la especificación de casos de uso, se puede tomaron las recomendaciones incluidas en [2] [3][14][22] y para el documento de especificación de requisitos de software, [8][17]).

Para esta actividad se propone la utilización de un nuevo diagrama al que se ha denominado "diagrama de precedencias", en el cual se muestran gráficamente las precondiciones de cada uno de los casos de uso que se incluyen en sus respectivas especificaciones. La idea de dicho diagrama fue tomada de Doug Rosengberg [20] quien propone la realización de un diagrama similar al propuesto, especificando las relaciones "precede" e "invoca" (invoke en inglés) para determinar los requerimientos de usuario. La Figura 1, muestra un ejemplo de un diagrama de precedencias.

El diagrama se utilizará para saber qué caso de uso necesita de alguna funcionalidad o información que es administrada o implementada por otro caso de uso. Esto permitirá conocer qué caso de uso debe programarse antes que otro, de manera que lo necesario para que se pueda implementar un caso de uso ya haya sido desarrollado en una iteración previa.

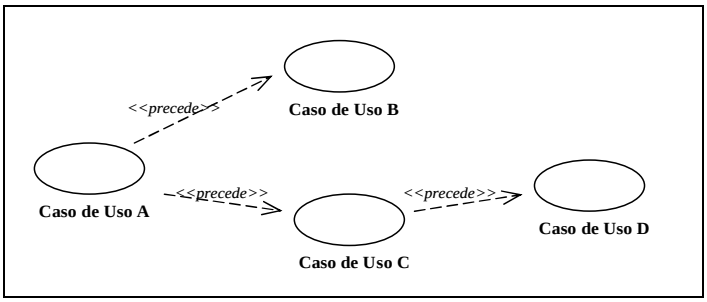


Figura 1. Ejemplo de Diagrama de Precedencias

Siguiendo la idea del párrafo anterior, los casos de uso que se encuentran a la izquierda del diagrama, se implementarán antes de los que se encuentran a la derecha; es decir, en el ejemplo propuesto en la Figura 1, "Caso de uso A" se deberá implementar antes que "Caso de uso C".
Es importante resaltar que en este diagrama no se consideran los casos de uso incluidos y extendidos, ya que éstos se pueden considerar como parte de aquellos que les hacen referencia.

3.2 Estimación del Esfuerzo para Cada Iteración (paso 2).
La siguiente actividad, después de determinar qué caso de uso se realizará en cada iteración, consiste en determinar el esfuerzo que tomará cada iteración. Para ello, se propone la utilización de las técnicas de puntos de función y COCOMO II.
La primera tarea consiste en determinar los puntos de función sin ajustar para cada iteración. El criterio seguido, para adaptar la técnica de puntos de función para ciclos de vida iterativos-incrementales, consiste en considerar que el resultado del cálculo de los puntos de función sin ajustar para todo el proyecto y sin considerar iteraciones (PFsA_Total), deberá ser igual a la suma de los puntos de función sin ajustar calculados para cada iteración por separado (***PFsA(i)*** = puntos de función sin ajustar de la iteración "i").

$$PFsA\_Total = \sum_{i=1}^{n} PFsA(i)$$

Uno de los aportes más importantes de esta propuesta consiste en determinar los puntos de función sin ajustar correspondientes a los ficheros lógicos internos (ILF) y ficheros de interfaz externa (EIF) para cada caso de uso. Para ello, esta técnica propone utilizar la siguiente fórmula (***Fich_PFsA*** = punto de función sin ajustar para un caso de uso "j" debido a los ficheros ILF/EIF, ***TCU(i)*** = número total de casos de uso que utiliza un ILF/ EIF "i", ***Peso(i)***= Peso debido a la complejidad del ILF/EIF "i", *i*= ILF/ EIF utilizado en el caso de uso "j" y *j*= caso de uso en cuestión)

$$Fich\_PFsA(j) = \sum_{i=1}^{n} \frac{1}{TCU(i)} xPeso(i)$$

Con los resultados obtenidos con la fórmula anterior y sabiendo qué caso de uso se desarrollará en cada iteración, se determinan los PFsA debido a los ficheros por iteración. A esto se le añadirán los PFsA correspondientes a las transacciones. Para ello, se utiliza la fórmula que se muestra a continuación (*i*= iteración específica, ***TPFsA(i)***= total de puntos de función sin ajustar para una iteración "*i*", ***Fich_PFsA(j)*** = punto de función sin ajustar para un caso de uso "*j*" debido a ILF/EIFs, ***Trans_PFsA(j)*** = punto de función sin ajustar para un caso de uso "*j*" debido a las transacciones (EI,EO,EQ) y *j*= caso de uso a implementar en una iteración "*i*").

$$TPFsA(i) = \sum_{j=1}^{n} \left[Fich\_PFsA(j) \right] + \sum_{j=1}^{n} \left[Trans\_PFsA(j) \right]$$

Luego, utilizando COCOMO II y con los puntos de función sin ajustar correspondientes a cada iteración, se obtiene el esfuerzo en meses-hombre de cada una de las iteraciones y, con este valor, se puede determinar el tiempo y personas necesarias para acabar la iteración y por ende el tiempo de todo el proyecto.
Es importante resaltar que el contexto del proyecto puede cambiar al pasar de una iteración a otra (conocimiento de la plataforma de desarrollo, integración del equipo de desarrollo, etc), por lo que podría ser necesario reestimar de nuevo el esfuerzo requerido para las iteraciones siguientes, revisando ficheros (ILF/EIF) y transacciones (EI, EO y EQ) en caso de cambio de requisitos, y recalculando drivers de coste propuestos por COCOMO II en caso de cambios contextuales u organizacionales.

4. DISEÑO DE LA EXPERIMENTACIÓN

Los estudiantes que participaron en esta experimentación fueron alumnos de cuarto año de la carrera de Ing. Informática de la Pontificia Universidad Católica del Perú. Esta experimentación se desarrolló en una asignatura obligatoria del área de Ingeniería de Software y como parte del curso los alumnos tenían que realizar un proyecto de desarrollo de software. La duración fue de 14 semanas correspondientes al semestre 2004-2 (agosto-diciembre 2004). El principal objetivo de esta experimentación fue el que los alumnos aplicaran la técnica de puntos de función para estimar el esfuerzo de la segunda y tercera iteración en base a su propio esfuerzo realizado en iteraciones previas. El objetivo secundario fue recolectar datos empíricos sobre el esfuerzo real necesario para cada iteración y observar los resultados de la aplicación de la técnica.

4.1. Los Estudiantes y la Distribución de los Equipos

Los estudiantes fueron divididos de manera aleatoria en equipos de 11 ó 12 personas. Para la conformación de equipos se siguieron dos criterios: igual proporción de alumnos con rendimiento académico similar entre los equipos e igual proporción de hombres y mujeres. Estos criterios fueron considerados, ya que empíricamente los profesores de la carrera han comprobado que estos factores influyen en el desempeño de los proyectos seguidos en los cursos.
La tabla 2 muestra los conocimientos y experiencias de los alumnos antes de comenzar el semestre en el que se realizó esta experimentación.

Tabla 2. Conocimientos y experiencias de los estudiantes al inicio del semestre

Característica	Conocimiento y/o Experiencia
Lenguajes y Entornos de Programación	• Java, C#, Pascal, C y Prolog. • No conocían Delphi.
Bases de Datos	• Oracle 8. • No conocían MS SQL Server.
Técnicas de análisis y diseño	• Estructurado y orientado a objetos
Gestión de proyectos	• Experiencia en proyectos pequeños de programación (equipos de 3 ó 4 estudiantes). • No conocían técnicas de planificación ni estimación.

Muchos de los cursos de la carrera de Ing. Informática de la PUCP consideran proyectos como parte aplicativa de los conceptos teóricos impartidos. Los alumnos antes de tomar el curso ya tenían experiencia en realizar trabajos grupales, aunque no conocían técnicas de planificación y estimación de proyectos de software.

4.2. Características del Proyecto

Cada equipo tenía que desarrollar un sistema para una cadena de tiendas por departamentos siguiendo la metodología RUP[17]. La duración del proyecto fue de 14 semanas que es la duración del semestre académico. Antes de realizar la fase de construcción, cada equipo tuvo que finalizar el documento de especificación de requisitos (ERS) [8] con casos de uso. La arquitectura del sistema se basó en el modelo cliente/servidor de dos capas [21] y fue validado mediante la realización de un prototipo.
Todos los equipos tenían que realizar tres iteraciones de la fase de construcción. Cada iteración duró dos semanas. La cantidad de casos de uso a implementar en la segunda y tercera iteración se basaba en el esfuerzo estimado en iteraciones previas.
A cada equipo se le asignó un asistente de docencia (al que se le conoce en la PUCP con el nombre de jefe de práctica) que se encargaba de asesorar a los estudiantes en el desarrollo del proyecto. Ellos apoyaron a los estudiantes para obtener un ERS que soporte los procesos del negocio de una verdadera cadena de tiendas por departamentos. Todos los asistentes de docencia trabajan en proyectos de desarrollo de software en la industria con una experiencia promedio de siete años.
La siguiente tabla muestra el software utilizado para el desarrollo del proyecto.

Tabla 3. Software utilizado en el proyecto

Tipo	Software
Herramienta de programación	Delphi 7.0
Sistema operativo	Microsoft Windows 2000
Base de Datos	Microsoft SQL Server 2000
Herramienta de modelado	Rational Rose
Herramienta para documentar	MS Office 2003

4.3. Recolección de Datos

Semanalmente los alumnos tenían que entregar de manera individual una hoja de cálculo en MS Excel, en el que incluían las horas trabajadas en cada actividad por día. La figura siguiente muestra un ejemplo de la hoja que utilizaron todos los integrantes de los equipos.

		Semana						6	Semana						7
	Actividad	26-09-2004	27-09-2004	28-09-2004	29-09-2004	30-09-2004	01-10-2004	02-10-2004	03-10-2004	04-10-2004	05-10-2004	06-10-2004	07-10-2004	08-10-2004	09-10-2004
1	Reuniones de coordinación (internas)														
2	Reuniones de control de proyecto (con JP y/o profesor)														
3	Elaboración de informes (no incluye ningún tipo de plan)														
4	Actividades de gestión de proyectos (edt, riesgos, gantt, plan del proyecto, plan de iteración).														
5	Determinación de requisitos (lista de exigencias , especificación casos uso).														
6	Determinación de la arquitectura del sistema														
7	Preparación del plan de pruebas.														
8	Diseño de interfaz de usuario									2	2	2	2	2	2
9	Diseño (no interfaz de usuario) y programación														
10	Pruebas de software.														
11	Integración de software.														
12	Elaboración de manual de usuario														
	TOTAL	0	0	0	0	0	0	0	0	2	2	2	2	2	2
		Acum. Semanal						0	Acum. Semanal						12

Figura 2. Ejemplo de Hoja para el Llenado de Horas Trabajadas.

Es importante mencionar que se les recalcó a los alumnos que la cantidad de horas que utilizaran en el proyecto no iba a influir en la calificación final del curso, que lo importante era que cumplieran con los casos de uso que se habían comprometido a hacer para cada iteración. Esto se hizo para asegurar la honestidad en el registro del esfuerzo realizado por cada uno de los integrantes de cada equipo.

Mediante reuniones con cada equipo, se verificó que el criterio para el llenado de las hojas sea el mismo para todos. Inicialmente hubo problemas, ya que habían alumnos que registraban más horas de lo que realmente utilizaban para el proyecto.

5. RESULTADOS OBTENIDOS.

En esta sección explicarán inicialmente los resultados obtenidos previamente al estudio realizado. Luego, se mostrarán los resultados obtenidos en el semestre 2004-2.

5.1 Resultados Obtenidos Previamente a Este Estudio.

Esta técnica propuesta ha sido usada en proyectos de software en los que participa una sola persona con resultados alentadores (un ejemplo de su aplicación se encuentra documentada en una tesis de máster [15]). Los resultados obtenidos nos animaron a probar la propuesta en nuevos proyectos, incluyendo aquellos en los que participan más de una persona.

Durante el semestre 2004-1 (marzo-junio 2004) y dentro del mismo curso de la carrera de Ing. Informática, se aplicó la técnica mostrada en la sección anterior, en un proyecto de software, pero con las siguientes variantes:

- A los estudiantes no se les explicó la técnica mencionada en la sección 3 de este artículo y se les dijo que se distribuyeran la carga de trabajo usando los puntos de función sin ajustar relacionados a las transacciones.
- La planificación para saber que construir en cada iteración la hicieron en base al diagrama de precedencias.
- No se les pidió que hicieran el cálculo del esfuerzo, tal y como lo indica la técnica mencionada en la sección 3 de este artículo, sólo registraron las horas que trabajaron para el proyecto.

Se conformaron dos equipos de trabajo y cada uno de ellos tenía que desarrollar un software para una cadena de restaurantes. Cada equipo era independiente y aunque el tema era el mismo, los ERS de ambos equipos fueron diferentes entre sí. Cada equipo tenía asignado un asistente de docencia (jefe de práctica) con mucha experiencia en el tema y, que además, trabajaba hace 3 ó 4 años en una empresa de desarrollo de software.

Los integrantes de los equipos habían trabajado de manera conjunta en proyectos de asignaturas de semestres anteriores, por lo que el contexto correspondiente a las relaciones interpersonales fue la misma en todas las iteraciones. Además, se pudo observar que el único cambio que se produjo en el contexto de cada una de las iteraciones fue el conocimiento de la herramienta de programación y la experiencia en el desarrollo de aplicaciones, (factor de coste LEXP y AEXP).
A manera de ejemplo, la tabla 4, muestra los resultados obtenidos en uno de los dos equipos que trabajaron en el curso. En [16] se puede encontrar información más detallada de este estudio.

Tabla 4. Resultados para el grupo A (2004-1)

Iteración de Constr.	Factor EAF COCOMO II	Esfuerzo Real*
1	1.46	4.01
2	0.80	2.52
3	0.80	2.64

*Medido en horas-hombre/PFsA

Con la información obtenida por los dos equipos, se hizo el cálculo del índice de correlación lineal, entre el factor EAF de COCOMO II y el esfuerzo real utilizado en cada iteración. Los resultados se muestran en la tabla siguiente.

Tabla 5: Coeficiente de correlación de regresión lineal de EAF vs. esfuerzo por PFsA

Equipo	Número de Integrantes	Coeficiente Correlación
A	11	0.999
B	10	0.999

En la tabla anterior se puede observar que existe una alta correlación entre el factor EAF de COCOMO II y el esfuerzo real utilizado. Esto sirvió para utilizar factores EAF similares para la siguiente iteración.

4.2 Resultados Obtenidos en esta Experimentación.
El tema del proyecto del semestre 2004-1 fue diferente al tema propuesto en el semestre 2004-2. Esto se hizo para evitar posibles copias de lo realizado en el semestre anterior (el primero fue un sistema para una cadena de restaurantes tipo fast-food y el segundo, un sistema para una cadena de tiendas por departamentos). Aunque fueron proyectos diferentes, ambos corresponden a un tipo de sistema de información, cuyos procesos de ingreso y salida de datos son similares. Es por ello, que los resultados en el semestre 2004-1 se pueden comparar con los resultados obtenidos en el semestre 2004-2.
Debido a los resultados alentadores mostrados en la sección anterior, se decidió probar la técnica en el siguiente semestre (2004-2), con la diferencia de que los esfuerzos de las iteraciones sean calculados en base al esfuerzo real obtenido por el propio equipo en iteraciones previas, en vez de utilizar COCOMO II.
Se conformaron tres equipos de trabajo. Los datos recopilados en uno ellos fueron considerados poco confiables debido a que se produjeron algunos problemas durante desarrollo del proyecto, entre los que se incluyen discrepancias entre sus miembros.
La tabla 6 muestra los resultados del equipo A (11 participantes) y la tabla 7 del equipo B (12 participantes).

Tabla 6. Resultados para el equipo A

Iteración de Construcción	PFsA	Esfuerzo Real (horas-hombre)
1	172,66	526
2	208,22	324
3	86,12	220
TOTALES	**467,00**	**1070**

Tabla 7. Resultados para el equipo B

Iteración de Construcción	PFsA	Esfuerzo Real (horas-hombre)
1	159,13	667
2	135,87	298
3	91,00	308
TOTALES	**386,00**	**1273**

Para la fase de construcción, se indicó que la repartición de la carga de trabajo por integrante se debería realizar utilizando los PFsA obtenidos de la técnica mostrada en la sección 3. Es importante resaltar, que los resultados de la técnica en mención fueron revisados antes que los alumnos lo apliquen en su proyecto, debido a que recién estaban aprendiendo la técnica de puntos de función.

En el país no existen personas certificadas en la técnica de Puntos de Función, por lo que no se pudieron validar los resultados, aunque sí se siguieron las recomendaciones e indicaciones del manual de Puntos de Función [23]. Además, una de las personas que revisó la documentación tiene dos años de experiencia en el tema y fue alumno del máster de Ing. de Software de la Universidad Politécnica de Madrid, en donde se les enseña y les exige la aplicación de la técnica en mención para el proyecto de fin de máster.
En los resultados obtenidos en las tablas anteriores, no se han considerado las horas correspondientes a las reuniones internas de coordinación (punto 1 de la figura 2) y las reuniones con el asistente de docencia o profesor (punto 2 de la figura 2). Esto se debe a que sólo se quería considerar el trabajo efectivo realizado para la construcción del software (diseño, programación y pruebas).
Para los resultados mostrados en las tablas 7 y 8 de esta sección, se ha utilizado la magnitud de error relativo (MRE), la cual se define como sigue (*y*=esfuerzo real, $\hat{y}$ =esfuerzo estimado) [6]

$$MRE = \frac{\left| y - \hat{y} \right|}{y}$$

Para el esfuerzo estimado de cada iteración se ha utilizado la siguiente fórmula (***esf_estimado(i)***=esfuerzo estimado para la iteración "i", ***EAF(i)***= factor EAF para la iteración *"i"* y ***esf_real(j)***=esfuerzo real de la iteración *"j"*)

$$esf\_estimado(i) = \frac{EAF(i)}{i-1} \times \sum_{j=1}^{i-1} \frac{esf\_real(j)}{EAF(j)}$$

Tabla 7. Resultados para el equipo A

Iter. Constr.	Factor EAF	Esfuerzo Real*	Esfuerzo Estimado*	MRE
1	1,46	3,05	-	-
2	0,80	1,56	1,67	7,28%
3	0,80	2,55	1,61	36,87%

*Medido en horas-hombre/PFsA

Tabla 8. Resultados para el equipo B

Iter. Constr.	Factor EAF	Esfuerzo Real*	Esfuerzo Estimado*	MRE
1	1,46	4,19	-	-
2	0,80	2,19	2,30	4,72%
3	0,80	3,38	2,25	33,67%

De manera similar a lo ocurrido en el semestre 2004-1 (ver tabla 4), se pudo observar que los cambios que se produjeron en el contexto de cada una de las iteraciones fue el conocimiento de la herramienta de programación y la experiencia en el desarrollo de aplicaciones, (factor de coste LEXP y AEXP). Es por eso que los factores EAF de las tablas 7 y 8 son los mismos que al de la tabla 6.
De los resultados obtenidos, se puede observar que el MRE de la segunda iteración es menor al 8% en ambos equipos, esto quiere decir que la técnica utilizada fue bastante acertada para esta iteración En cambio, en la tercera iteración, se puede observar que el MRE es mayor al 33%, lo que podría suponer que la técnica falló para esta iteración. Inicialmente, se pensó que hubo un error en la recopilación de datos, pero luego de conversar con los alumnos hubo un detalle que no se tomó en cuenta. Para la última entrega, los alumnos tenían que presentar su proyecto a todos sus compañeros del curso, a todos los asistentes de docencia y al profesor, por lo que la presión de presentar un buen trabajo fue muchísimo mayor que en las otras iteraciones. Esto provocó que los equipos hicieran pruebas más intensivas del software e incluso modificaron partes que ya habían sido probadas y que funcionaban de manera correcta.
Para este problema se puede considerar lo que propone COCOMO II [4] que incluye un modelo para estimar el porcentaje de tamaño de software que se modificará de las iteraciones anteriores. A diferencia de la segunda iteración en la que no hubo problemas en el cálculo, ya que los alumnos no modificaron la funcionalidad hecha en la primera iteración.
En la siguiente tabla se muestra el porcentaje de modificación realizada de los puntos de función de la primera y segunda iteración, en la tercera iteración

Tabla 9. Porcentaje de uso de PFsA de 1ra y 2da iteración en la 3ra iteración.

Equipo	PFsA 1ra & 2da iteración*	PFsA 3ra iteración*	Total PFsA Usadas 3ra iteración**	% de Uso de PFsA de la 1ra y 2da Iteración
A	380.88	86.12	137.79	13.21%
B	295.00	91.00	138.56	15.66%

* PFsA tomado de la técnica propuesta
** PFsA calculados en base al esfuerzo estimado para la tercera iteración y el esfuerzo real realizado en esa iteración.

En la tabla anterior se puede observar que el porcentaje de modificación realizada de los PFsA de la primera y segunda iteración, en la tercera fue de aproximadamente 14%. Este dato será utilizado en futuras experiencias con alumnos.

6. DISCUSIÓN DE LOS RESULTADOS OBTENIDOS

Aunque la información obtenida corresponde a dos proyectos y los resultados obtenidos son alentadores para la técnica propuesta, aún no se puede afirmar que la técnica sea realmente confiable para otros contextos. Algunos de los comentarios de los alumnos con respecto a la técnica fueron los siguientes:

- La técnica les sirvió para medir el tamaño de la parte del software que tenían que implementar cada integrante para una iteración.
- El esfuerzo registrado en la primera iteración fue muy útil para calcular el esfuerzo de la segunda iteración.

También, se puede observar (ver la tabla 10) que el esfuerzo utilizado de ambos equipos es diferente. Por ejemplo, en la segunda iteración, el equipo A requirió de 1.56 horas-hombre/PFsA, en cambio el equipo B requirió 2.19 horas-hombre/PFsA. Algo similar ocurre para las otras iteraciones.

Tabla 10. Esfuerzo real para cada iteración por equipo

Equipo	Esfuerzo Real 1ra Iteración*	Esfuerzo Real 2da Iteración	Esfuerzo Real 3ra Iteración*
A (11 alumnos)	3.05	1.56	2.55
B (12 alumnos)	4.19	2.19	3.88

*Medido en horas-hombre/PFsA

La variación que se muestra en la tabla anterior se debe a muchos factores, alguno de ellos son: la cantidad de alumnos por equipo y la capacidad de trabajo en equipo, siendo algunos de ellos difíciles de cuantificar. Lo que sí se pudo comprobar es que la información del esfuerzo real utilizado es muy útil para la estimación del esfuerzo del equipo a quien le pertenecen esos datos.

En cuanto a los resultados obtenidos en la tercera iteración, el MRE fue aproximadamente al 35% para todos los proyectos y equipos (tablas 7 y 8), como se comentó esto se debía a los cambios que realizaron los primeros incrementos, debido a la presión producida por la presentación final. En la tabla 9 se puede mostrar que el porcentaje de modificación realizada de la primera y segunda iteración, en la tercera fue de aproximadamente 14%. Para ambas técnicas el porcentaje es muy similar. Este dato será utilizado en futuras experiencias con alumnos.

Es difícil hacer generalizaciones de los resultados obtenidos para otros contextos que no sea el académico. La ventaja de realizar experimentaciones con alumnos es el que se puedan controlar algunos factores o variables que no se pueden controlar en proyectos en la industria. Algunos de ellos son los siguientes:

- Conocimiento y experiencia de los miembros de los equipos de desarrollo
  En proyectos de la industria es muy difícil encontrar que la experiencia de todos los miembros sea la misma. En algunos casos, los miembros no tienen los conocimientos necesarios sobre desarrollo orientado a objetos.
- Rotación de personal
  En la industria es bastante frecuente el cambio de los integrantes de un proyecto de software por diversos factores, entre ellos, la asignación a otros proyectos. Para la experimentación, este factor no se produjo.
- Cambio de requisitos
  El cambio de requisitos es algo que ocurre frecuentemente en la industria. Este factor se puede controlar en un proyecto académico.
- Secuencia de implementación de los requisitos
  En esta experimentación se ha realizado la implementación de los requisitos siguiendo el diagrama de precedencias, lo cual a veces no se puede hacer realizar en la industria. Esto se debe a que la precedencia de construcción de los requisitos depende de los requerimientos de los usuarios y de los clientes.

Los resultados obtenidos se podrían aplicar en la industria cuando el equipo de trabajo cuente mucha experiencia en el desarrollo de aplicaciones y tenga que afrontar un proyecto en el que se tenga que utilizar una nueva herramienta y lenguaje de programación.

7. CONCLUSIONES Y TRABAJO FUTURO

La mayoría de las aproximaciones actuales para estimar proyectos, aún definiéndose como independientes de la tecnología y modelos de ciclo de vida, tienen un carácter fuertemente influido por los ciclos de vida en cascada. A pesar de que son válidas para otras aproximaciones, como los ciclos de vida iterativos-incrementales, por ejemplo, en general no ofrecen ninguna guía para acometer dicha adaptación.

El presente trabajo muestra una técnica que se basa en Puntos de Función para la estimación del esfuerzo en proyectos que utilizan un ciclo de vida iterativo-incremental. Además, se muestran los resultados obtenidos en proyectos de software realizados por estudiantes. La ventaja de utilizar alumnos es que se pueden controlar factores que podrían afectar el estudio, como por ejemplo, conocimiento y habilidades de los participantes y contexto de cada iteración.

Los resultados obtenidos son bastante alentadores, ya que la técnica propuesta da un error relativo entre el esfuerzo estimado y esfuerzo real menor al 10%. Los equipos utilizaron los datos de las iteraciones previas para hacer el cálculo del esfuerzo estimado para las siguientes iteraciones. Estos resultados no son concluyentes, ya que se necesitan realizar más pruebas, no sólo con alumnos sino también en la industria.

El trabajo futuro que está relacionado a este artículo es el siguiente:

- Adaptar la técnica propuesta de manera que los ficheros sean reemplazados por diagramas de clases, para que se adapte al enfoque orientado a objetos.
- Realizar experiencias en las que no se puede seguir la implementación con el orden sugerido por el diagrama de precedencias, ya que lo que muchas veces ocurre que la prioridad de implementación requerida en proyectos en la industria es muy diferente.
- Adaptar la técnica propuesta a otras técnicas como COSMIC-FFP y Puntos de Casos de Uso.
- Realizar experiencias con los alumnos para comparar la efectividad de otras técnicas, como COSMIC-FFP y Puntos de Casos de Uso, para estimar el esfuerzo de las iteraciones en proyectos de software.

Agradecimientos

Agradezco a la Dra. Ana M. Moreno, profesora de la Universidad Politécnica de Madrid, por sus comentarios a la versión inicial de este artículo. Asimismo, mi reconocimiento a todos los alumnos que participaron en esta experimentación y cuyos nombres de equipo fueron: CSM, AISBER y 11-Son-Suficientes.

Referencias

[1] Albrecht, A. J. *Measuring Application Development Productivity*, IBM Applications Development Symposium, Monterey, CA, USA, 1979.

[2] Bittner, K., *Use Case Modeling*, Addison-Wesley, USA, 2003.

[3] Bittner, K., "Why Use Cases Are Not Functions", http://www.therationaledge.com, USA, 2000.

[4] Boehm, B., et al. *Software cost estimation with COCOMO II*, Prentice-Hall, USA, 2000.

[5] Carver, J., Jaccheri, L., Morasca, S., *Issues in Using Empirical Studies in Software Engineering Education*, Proceedings METRICS'03, IEEE Computer Society, USA, 2003.

[6] Conte SD, HE Dunsmore, and VY Shen, *Software Engineering Metrics and Model*s, Benjamin-Cummings, Menlo Park CA, 1986.

[7] Fetcke, T., Bran, A., Nguyen, T., Mapping the OO-Jacobson Approach into Function Point Analysis. Proceedings of TOOLS-23'97, IEEE, USA, 1997.

[8] IEEE Computer Society, IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications, The Institute of Electrical and Electronics Engineers, USA, 1998.

[9] Jacobson, I., Object-Oriented Software Engineering. A Use Case Driven Approach, Addison-Wesley, USA, 1992.

[10] Karner, G. ***Metrics for Objectory. Diploma thesis***, University of Linköping, Sweden. No. LiTH-IDA-Ex-9344:21. December 1993.

[11] Longstreet, D.,Use Case and Function Points, http://www.softwaremetrics.com/, Longstreet Consulting Inc, USA, 2001.

[12] Object Management Group, OMG Unified Modeling Language, http://www.uml.org, USA, 1999.

[13] Pow-Sang, J., Imbert R., Estimación y Planificación de Proyectos Software con Ciclo de Vida Iterativo-Incremental y empleo de Casos de Uso, Proceedings IDEAS 2004, Arequipa-Perú, 2004.

[14] Pow-Sang, J., La Especificación de Requisitos con Casos de Uso: Buenas y Malas Prácticas, II Simposio Internacional de Sistemas de Información e Ing. de Software en la Sociedad del Conocimiento-SISOFT 2003, Pontificia Universidad Católica del Perú, Lima-Perú, 2003.

[15] Pow-Sang, J., GESPROMET, Sistema para la Gestión de Proyectos de Software Utilizando MÉTRICA Versión 3. Tesis de Máster en Ingeniería del Software, Universidad Politécnica de Madrid, España, 2002.

[16] Pow-Sang, J., ***Estudio Comparativo de Técnicas para la Estimación del Esfuerzo de las Iteraciones de Proyectos Software***, Proceedings JIISIC'04, Madrid-España, 2004.

[17] Rational Software, Rational Unified Process version 2001A.04.00.13, USA, 2001.

[18] Royce, W. W., Managing the Development of Large Software Systems: Concepts and Techniques. Proceedings WESCON, 1970.

[19] Rumbaugh, I. Jacobson, and G. Booch, Unified ModellingLanguage Reference Manual, Addison Wesley, 1997.

[20] Rosenberg, D., Scott, K., Use Case Driven Object Modeling with UML, Addison-Wesley, Massachusets, USA, 1999.

[21] Sadoski, D., Two Tier Software Architectures, http://www.sei.cmu.edu/str/descriptions/twotier.html, SEI, USA, 2004.

[22]Schneider, G. and Winters, J. ***Applying Use Cases*** – ***A Practical Guide***, 2nd Edition. Addison-Wesley. USA, 2001.

[23] The International Function Point User Group (IFPUG), Function Point Counting Practices Manual-Release 4.1, USA, 1999.

Validating Dynamic Software Architectures using Alloy

Nazareno Aguirre, María Marta Novaira and Sonia Permigiani
Departamento de Computación - FCEFQyN
Universidad Nacional de Río Cuarto
Ruta 36 Km. 601, Río Cuarto (5800), Córdoba, Argentina
{naguirre, mnovaira, spermigiani}@dc.exa.unrc.edu.ar

Abstract

In this paper we show how the Alloy specification language, a formal language whose semantics is based on the mathematical notion of relation, can be used in order to model and analyse dynamic software architectures. We show how fundamental notions of software architectures, such as components, connectors and configurations, can be specified in Alloy. Moreover, we also discuss the possibility of describing and analysing static and dynamic properties of software architectures in Alloy. We enumerate some shortcomings of Alloy as a language for expressing dynamic properties of software architectures, and argue for the use of an extension of Alloy, called DynAlloy, for this task.

1 Introduction

It is widely accepted that, due to the complexity of modern software systems, it becomes necessary to put emphasis in the design and decomposition of systems in terms of interacting components or modules, in the process of software construction. The field of software architectures [7] encourages the description and design of systems at a high level of abstraction, in terms of *components*, which represent the computational units of the system, interconnected by means of *connectors*, which represent the interaction mechanisms. A variety of languages, called architecture description languages, have been proposed in order to model software architectures. Some of these are Acme [6], Darwin [13], C2 [15], Rapide [12], CommUnity [17], and others. Some of these languages have associated tool support, that allow one to perform different kinds of analysis activities, such as simulation, checking of integrity constraints, consistency verification, etc. Some of these architecture description languages are able to deal with an important feature, namely the so called dynamic, i.e., run time, architectural changes [14]. Dynamic architectural reconfiguration is commonly found in the design of software systems, perhaps motivated by object orientation, where some dynamic reconfiguration (e.g., dynamic creation and deletion of modules) is intrinsic. Although many tools support some analysis regarding software architectures (in some of the above mentioned languages), little tool support for analysis is available when specifying dynamic software architectures, i.e., software architectures that present dynamic architectural changes. We propose here the use of a formal notation called Alloy for describing and analysing dynamic software architectures.

Alloy is a formal specification language with a clear semantics, based on the well known mathematical notion of relation [8, 10]. The Alloy language has recently gained interest in the formal methods community, mainly due to its simplicity and the fact that it has been designed with the goal of making specifications automatically analysable. Indeed, Alloy counts on a tool for automatic analysis of specifications, the Alloy Analyzer. Given a software system specification and a property to be validated, the Alloy Analyzer allows one to search for models of the specification which are counterexamples of the property being validated. Since first-order logic is not decidable, and Alloy's underlying formalism is a proper extension of first-order logic, this search for counterexamples cannot be exhaustive, and has to be performed up to a certain bound k on the number of elements in the universe of the models. Nevertheless, this counterexample extraction mechanism seems to be useful in practice, since as it has been observed by D. Jackson, if a specification has counterexamples, it often has counterexamples of small size [10].

In this paper, we propose the use of Alloy, and the Alloy Analyzer, for validating properties of software architectures. We show how some fundamental concepts of software architectures, such as components, connectors and configurations, can be straightfowardly modelled in Alloy, and how the Alloy Analyzer can

be employed to validate properties of the specified software architectures. In addition, we discuss some limitations of the Alloy language and its associated tool support for describing and analysing *dynamic* properties of software architectures, i.e., properties regarding *executions* of the specified architectures. We observe some limitations of the Alloy language for this task, and argue for the use of a, to our understanding, suitable extension of Alloy, called DynAlloy [5].

2 Modelling Software Architectures

We present here, briefly and by means of an example, how software architectures can be modelled in Alloy. In the software architectures field, software systems are typically modelled at a high level of abstraction, using the abstract notions of component and connector [7]. Components represent the computational units of a system, and can be associated with classes, modules, subsystems, etc, whereas connectors represent the interactions between these components. A fundamental characteristic of connectors, as opposed to other forms of representing communication between components, is that the interaction is defined externally to the definition of the components.

2.1 Modelling Components

Alloy features a construction we can take advantage of for the definition of components: the signature. Signatures can be used to define data domains, as well as components with internal structure. Let us consider, for example, that we need to model the well known architectural pattern called Producer-Consumer. We might start by saying that we will need to model the elements that the producers produce and the consumers consume, i.e. the data that is exchanged between producers and consumers. Moreover, in case we require various instances of producers and consumers for particular scenarios, we could also think about equipping producers and consumers with a name.

Data and names of components can be modelled by using Alloy's basic signatures (i.e., signatures without internal structure):

```
sig Data {}
```

```
sig Name {}
```

Producers and consumers can also be modelled by using Alloy's signatures, but unlike names and data, these will have an internal structure:

```
sig Producer {
    n: Name,
    queue: set Data,
    outgoing: Data
}

sig Consumer {
    n: Name,
    consumed: set Data,
    incoming: Data
}
```

Note that signatures denote the structure of the producers and consumers, and not a particular producer or consumer (essentially, they describe the *type* of consumers and producers, and, as we will show, we will be able to have several different instances of these components). A producer has a name (the field or attribute named `n`), a queue modelled by a set of data (the field `queue`) and an element ready to be sent (the field `outgoing`). A consumer consists of a name (again, the field named `n`), a set of elements that have been already consumed (modelled by the field `consumed`), and a data element, ready to be consumed (the field `incoming`). As a remark, notice that both `Producer` and `Consumer` are "reference free", in the sense that their definitions do not make use of reference variables to implement communication. This is so because the model of interaction between components will be based on action synchronisation and shared variables, as in CommUnity [2, 17]. The communication between components will be defined externally to the specification of components, by means of *connectors*, and respecting the software architectures philosophy.

2.1.1 Specifying Services of Components

The components of the previous section encapsulate information, modelled by the fields of the signatures. However, these do not include behaviour. We can model behaviour, i.e., operations or services of the components, by means of *functions* in Alloy. For instance, we might consider operations for producing and sending data in a producer, and for consuming data in a consumer:

```
fun produce ( p, p' : Producer, d: Data ){
  p'.n = p.n
  p'.queue = p.queue + d
  p'.outgoing = p.outgoing
}

fun send ( p, p' : Producer, d: Data ){
  d in p.queue
  p'.n = p.n
  p'.queue = p.queue - d
  p'.outgoing = d
}


fun receive ( c, c' : Consumer, d: Data ){
  c'.n = c.n
  c'.consumed = c.consumed + c.incoming
  c'.incoming = d
}
```

Note that, in the style of Z specifications [16], the state of a component after the execution of an operation is captured by the values of primed variables in function definitions. However, as observed in [5], this is just a convention, not reflected in the semantics of Alloy functions. As the reader might appreciate, the meaning of these function definitions is rather easy to understand, thanks to the simplicity of Alloy's syntax.

2.2 Modelling Systems and Connectors

Since Alloy is based on relations, and signatures can be composed of relation typed fields, we can model connectors by means of relations. In the example developed here, we will only make use of binary connectors, although more complex connectors might also be characterised. Suppose we want to represent a system consisting of varying numbers of producers and consumers; moreover, we want these components to communicate. So, we need to represent the connections between (live) producers and (live) consumers. Then, a system of communicating producers and consumers would consist of a set of live producers, a set of live consumers, and a binary relation between these:

```
sig System {
   ccs: set Consumer,
   pps: set Producer,
   conn: pps -> ccs
}
```

2.2.1 Modelling Operations of the System

Now that we have defined the system, we can use Alloy's functions to model operations of the system. Unlike other operations, for instance those of producers and consumers, we can define the operations of the system in an incremental way, in the sense that new operations can be defined in terms of already defined ones. A particular kind of system operation that can be defined is that of *reconfiguration operations*, i.e., operations

that might modify the architectural structure of the system at run time. An example of such an operation could be one that creates a new live producer. The specification of this operation is the following:

```
fun sysnewprod (s, s' : System, p: Producer ){
 s'.pps = s.pps + p
 s'.ccs = s.ccs
 s'.conn = s.conn
}
```

Other reconfiguration operations, rather important for our example, are the ones that allow us to connect or disconnect producers with consumers dynamically. Consider, for instance, the operation that connects a live producer with a live consumer:

```
fun connect(s,s': System, p: Producer, c: Consumer) {
 p in s.pps && c in s.ccs
 s'.pps = s.pps
 s'.ccs = s.ccs
 s'.conn = s.conn + (p -> c)
}
```

2.3 Functions Associated with Connector Definitions

In the definition of the System signature, we represented connectors by means of a relation between producers and consumers. But this does not model the interaction between the instances. We can define the interaction, related to the existence of a connection between components (represented by an ordered pair in the `conn` relation, in our example) by means of functions of System:

```
fun syssend(s, s' : System, p,p':Producer, c,c':Consumer, d: Data) {
 p in s.pps && c in s.ccs &&  s'.conn = s.conn && (p -> c) in s.conn
 send(p,p',d)
 receive(c,c',d)
 p.n = p'.n
 c.n = c'.n
 p' in s'.pps && c' in s'.ccs
}
```

Notice that, as part of the precondition of this function, we have that the ordered pair `(p->c)` must belong to `s.conn`.

2.4 Integrity Constraints

We can impose integrity constraints to the specified software architecture by means of Alloy's *facts*. Due to some limitations of the Alloy specification language, one can use facts to impose only *static* restrictions, i.e., restrictions that refer to structural properties, but *not* properties of executions. We will see later on that there exist mechanisms for characterising properties of execution traces, but this requires a complicated specification of traces within the model of the architecture [5].

As a example, consider the restriction that enforces producers and consumers not to share their name spaces. This can be easily expressed in Alloy as follows:

```
fact NoSharedNamesPC  {
 all s: System | all p : s.pps | all c : s.ccs | ! (p.n = c.n)
}
```

We might also require that different producers (resp. consumers) are assigned different names:

```
fact NoSharedNamesP  {
 all s: System | all p1,p2 : s.pps | p1.n = p2.n => p1 = p2
}

fact NoSharedNamesC  {
 all s: System | all c1,c2 : s.ccs | c1.n = c2.n => c1 = c2
}
```

2.5 Predicates About the States of Systems

Sometimes is useful to exploit Alloy's functions for capturing special states of systems. This is of special interest for the validation activities, as we will see later on in this paper. As an example, let us suppose that we want to characterise those systems whose underlying connector `conn` is *functional*:

```
fun functionality_conn(s : System) {
   all p1,p2 :  s.pps |
   all c1,c2 : s.ccs |
  (p1 = p2) && c1 in p1.(s.conn) && c2 in p2.(s.conn) => c1 = c2
}
```

3 Validating Properties of Architectural Specifications

In Alloy, the intended properties of a model are specified as *assertions*. One can then employ the Alloy Analyzer to validate the specification, i.e., to verify, for bounded models, that the assertions are consequences of the model and its facts. Let us start by considering some static structural properties of dynamic software architectures.

3.1 Validating Static Properties

In this section we consider some static intended properties of dynamic software architectures that one might want to validate. To start with, let us consider a simple assertion. Suppose we want to check whether operation `sysnewprod` expands the set of live producers. We can check that property by means of the following assertion in Alloy:

```
assert pps_expansion {
 all s,s': System |
 all p : Producer |
         sysnewprod(s,s',p) => s.pps = s'.pps - p
}
```

It turns out that this assertion is false, and the Alloy Analyzer quickly finds a counterexample of it. The problem is that we would need to add that `!(p in p.pps)` as a precondition of `sysnewprod(s,s',p)`. Notice that the validation of this assertion refers to a reconfiguration operation, although is not a property of execution traces.

Another simple property to validate could be that the operations `sysnewprod` and `connect` preserve the functionality of the connector `conn`. We can assert these properties in the following way:

```
assert sysnewprod_func_preservation {
 all s,s': System |
```

```
 all p : Producer |
         sysnewprod(s,s',p) && functionality_conn(s) =>
         functionality_conn(s')
}

assert conn_func_preservation {
 all s,s': System |
 all p : Producer | all c : Consumer |
 !(p in s.pps) && !(c in s.ccs) &&
 connect(s,s',p,c) && functionality_conn(s) =>
         functionality_conn(s')
}
```

Both these assertions are valid properties of the specification. The second one, however, holds because the antecedent of the implication does never hold. We will use variants of these assertions for validation with respect to execution traces.

3.2 Validating Properties of Execution Traces

As it has been observed in [3, 5], Alloy is not appropriate for the validation of properties regarding execution traces of systems. These properties are of fundamental importance, especially in dynamic architectures. Although Alloy is limited in this sense, in [9], the authors propose a mechanism for representing and validating, by means of the Alloy Analyzer and within standard Alloy, properties of traces. This mechanism consists, essentially, of complementing the specification of a system with an explicit specification of traces. A main drawback of this mechanism is that the definition of execution traces strongly depends on the property of traces that one would want to validate. For example, if we would like to check whether sequences of applications of `connect` and `sysnewprod` preserve the functionality of `conn`, then system traces would be defined as follows:

```
sig Tick {}

sig SystemTrace {
 ticks: set Tick,
 first, last :ticks,
 next: (ticks-last) ! -> !(ticks-first),
 state: ticks ->! System
}
{
 first.*next=ticks
 all t:ticks - last |
   some s = t.state, s' =t.next.state |
   some p: Producer, c:Consumer | connect (s,s',p,c)
   || (some p: Producer| sysnewprod(s,s',p))
}
```

We might then attempt to validate that sequences of applications of `connect` and `sysnewprod` preserve the functionality of `conn` by saying that, for every trace, if `conn` is functional in the initial state, then it is also functional in any (initial, intermediate or final) state of the trace. This is easily specified as an Alloy assertion in the following way:

```
assert trace_func_preservation {
    all t: SystemTrace | functionality_conn((t.first).(t.state)) =>
    all s: Tick | s in t.ticks => functionality_conn(s.(t.state))
}
```

This is, in fact, a valid assertion, and therefore the Alloy Analyzer will fail to find counterexamples for any bound imposed in the size of the domains.

3.3 Validating Properties of Executions in DynAlloy

As it has been observed in [5], in order to validate properties of executions in Alloy it is required the specifier to provide a specification of traces, as the one given above. This is not a good engineering practice, since execution traces should not be specified explicitly, but they should get determined by the combination of operations one needs to analyse. Therefore, an alternative is provided in [5], which basically replaces functions as operation descriptions by *actions*, which can be combined to form *programs* by means of nondeterministic choice, sequential composition, iteration, etc.

In a DynAlloy approach, one specifies basic operations as actions, via partial correctness assertions. So, operations such as `produce` or `send`, are expressed as follows:

```
act produce(p: Producer, d: Data) {
    pre { }
    pos { (p'.n = p.n) and (p'.queue = p.queue + d) and (p'.outgoing = p.outgoing) }
}

act send(p: Producer, d: Data) {
    pre { d in p.queue }
    pos { (p'.n = p.n) and (p'.queue = p.queue - d) and (p'.outgoing = d) }
}
```

Other, more complex actions, are specified likewise:

```
  act sysnewprod ( s : System, p: Producer ) {
   pre { }
   pos {  s'.ccs = s.ccs and
    s'.conn = s.conn and
    s'.pps = s.pps + p
   }
  }
```

Notice that primed variables are not necessary anymore in the definition of operations, since actions have an input/output meaning, explicitly reflected in the semantics. Moreover, one can combine these basic actions to form more complex ones, as for instance, one can define the above operation `syssend` as a program:

```
  syssend(s, p, c, d) = send(p,d) ; receive(c,d)
```

Note that we have lost the pre- and post-condition specification of this action; one can write an assertion to validate that this program behaves as intended, in the following way:

```
assert syssendSemantics {
    assertCorrectness {
        pre = { some s : System | p in s.pps and c in s.ccs and (p -> c) in s.conn  }
        prg = { send(p,d) ; receive(c,d) }
        pos = { p'.n = p.n and p'.queue = p.queue - d and p'.outgoing = d and
              c'.n = c.n and c'.consumed = c.consumed + c.incoming and c'.incoming = d }
    }
}
```

The specification of some properties of traces, such as the one shown in the previous section, regarding the preservation of functionality of `conn`, is easier to specify in DynAlloy. The corresponding specification is the following:

```
act connect(s: System, p: Producer, c: Consumer) {
        pre { p in s.pps and c in s.ccs }
        pos { s'.pps = s.pps and
              s'.ccs = s.ccs and
              s'.conn = s.conn + (p -> c) }
}

act sysnewprod ( s : System, p: Producer ) {
        pre { }
        pos { s'.ccs = s.ccs and
              s'.conn = s.conn and
              s'.pps = s.pps + p }
}

assert invariance_functionality_conn {
    assertCorrectness {
        pre = { functionality_conn(s) }
        prg = { (sysnewprod(s,p)+connect(s,p,c))* }
        pos = { functionality_conn(s') }
   }
}
```

We had to rewrite the specification of some of the operations, which now need to be characterised by means of DynAlloy's actions. As it can be seen here, the specification is much clearer than the standard Alloy equivalent. Moreover, if we need to validate other properties of traces, corresponding to different programs, we only need to write new partial correctness assertions. But, we will not need to provide, as is the case with standard Alloy, different definitions for traces. As described in [5], it is even more efficient to check for the validity of these kinds of programs (i.e., those involving nondeterministic choice in the scope of iteration) in DynAlloy.

However, DynAlloy has an important limitation with respect to the Alloy approach to the specification of properties of traces. DynAlloy allows one to validate an important class of properties of traces, the so called *invariance properties*. But, in its current form, DynAlloy cannot deal with more general properties of traces. Consider, as an example, the following assertion regarding execution traces in Alloy:

```
fun total_conn(s: System) {
    all p : s.pps | some c : Consumer | (p -> c) in s.conn
}

assert eventual_totality_conn {
    all t : SystemTrace | some i : Tick | i in t.ticks &&  total_conn(i.(t.state))
}
```

This assertion is not expressible in DynAlloy by means of actions and implicit traces. However, it is well known in the area of concurrent systems that only infinite traces can be actual counterexamples of liveness properties (liveness properties do not exclude finite prefixes of runs) [1]. Since (Dyn)Alloy's analysis mechanism cannot handle infinite runs, neither the DynAlloy tool nor the Alloy Analyzer can be employed in order to *validate* progress properties. Nevertheless, one could take advantage of the Alloy characterisation of properties of traces, and attempt to use a proof calculus, such as that presented in [4], to perform deductive reasoning regarding liveness.

As a last example, let us consider a variant of the above assertion `eventual_totality_conn`. Suppose that we would like to check that, in traces of length at least $(n \times 2)$, where n is the number of producers, the assertion `eventual_totality_conn` holds (recall that traces correspond, for this example, to sequences of applications of operations `connect` and `sysnewprod`). This can be captured in standard Alloy by adding, for a particular n, a fact expressing that there are at least n different ticks in each trace (this is straightforwardly expressed in first-order logic with equality), and then checking the assertion for a bound of $(n \times 2)$. Checking this assertion fails, since one needs to impose some further conditions, namely:

- that operation `connect` "connects" only disconnected producers; that is, we have to replace the definition of `connect` by the following:

```
fun connect(s,s': System, p: Producer, c: Consumer) {
        !(some c1 : Consumer | (p -> c1) in s.conn)
        p in s.pps
        c in s.ccs
        s'.pps = s.pps
        s'.ccs = s.ccs
        s'.conn = s.conn ++ (p -> c)
}
```

- that operation `sysnewprod` effectively adds new producers, i.e., the added producer was not already live:

```
fun sysnewprod ( s, s' : System, p: Producer ) {
        !(p in s.pps)
        s'.ccs = s.ccs
        s'.conn = s.conn
        s'.pps = s.pps + p
}
```

4 Conclusions and Future Work

We have developed a simple case study, in order to assess the modelling and analysis of dynamic software architectures in Alloy. We presented the case study with some degree of detail, the characterisation of the essential constructions in software architectures, and some ideas regarding how to model and analyse dynamism in Alloy. We compared the use of Alloy and an extension of it, called DynAlloy [5], for the task of specifying and validating properties of executions. Although DynAlloy is better suited for the specification of properties of traces (implicitly and by means of program definitions), the language is less expressive than the standard Alloy approach to the characterisation of executions. We therefore observe that we need to combine both approaches, applying the DynAlloy one whenever possible, and using Alloy's explicit trace definitions when DynAlloy is not applicable. We also notice that none of the approaches, however, can be used to validate liveness properties.

As work in progress, we are currently studying forms of representing more complex properties of execution traces in DynAlloy. For this task, we are experimenting with extensions to DynAlloy to cope with general dynamic logic formulas.

Since the Alloy Analyzer can only be used to validate properties (as opposed to verifying properties), we are also exploring the use of the complete proof calculus presented in [4] to perform deductive verification of assertions in Alloy specifications. We are examining a characterisation of a similar proof calculus for a related language in PVS, presented in [11], for this task.

We also plan to provide a translation from the architecture description language Acme into (Dyn)Alloy, so that the software architect can use a more familiar front end for expressing his specifications, and the properties to be validated. We believe that the use of the (Dyn)Alloy language can contribute in the analysis

of properties of dynamic software architectures, complementing the tool support provided for some particular architecture description languages with dynamic reconfiguration.

References

[1] B. Alpern and F. Schneider, *Defining Liveness*, Information Processing Letters 21(4): 181-185, 1985.

[2] J. Fiadeiro and T. Maibaum, *Categorical Semantics of Parallel Program Design*, Science of Computer Programming, Vol. 28, issue 2-3, Elsevier, 1997.

[3] M. Frias, C. López Pombo, G. Baum, N. Aguirre and T. Maibaum, *Taking Alloy to the Movies*, in Proceedings of the International Symposium on Formal Methods FM 2003, Pisa, Italy, Lecture Notes in Computer Science, Springer-Verlag, 2003.

[4] M. Frias, C. López Pombo and N. Aguirre, *An Equational Calculus for Alloy*, in Proceedings of the International Conference on Formal Engineering Methods ICFEM 2004, Seattle, USA, Lecture Notes in Computer Science, Springer-Verlag, 2004.

[5] M. Frias, J. Galeotti, C. López Pombo and N. Aguirre, *DynAlloy: Upgrading Alloy with Actions*, in Proceedings of the International Conference on Software Engineering ICSE 2005, St. Louis, USA, ACM Press, 2005.

[6] D. Garlan and R. Monroe and D. Wile, *ACME: An Architecture Description Interchange Language*, in Proceedings of CASCON'97, Toronto, Ontario, 1997.

[7] D. Garlan and M. Shaw, *An Introduction to Software Architecture*, in V. Ambriola and G. Tortora (eds), Advances in Software Engineering and Knowledge Engineering, Series on Software Engineering and Knowledge Engineering, Vol 2, World Scientific Publishing Company, 1993.

[8] D. Jackson, *Lightweight Formal Methods*, in Proceedings of the International Symposium on Formal Methods Europe FME 2001, Berlin, Germany, Lecture Notes in Computer Science, Springer-Verlag, 2001.

[9] D. Jackson, I. Shlyakhter and M. Sridharan, *A Micromodularity Mechanism*, in Proceedings of the 8th European Software Engineering Conference held jointly with the 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering ESEC/FSE 2001, Viena, Austria, 2001.

[10] D. Jackson, *Alloy: a lightweight object modelling notation*, in ACM Transactions on Software Engineering and Methodology (ACM TOSEM), Vol. 11, Nro. 2, 2002.

[11] C. López Pombo, S. Owre and N. Shankar, *A Semantic Embedding of the Ag Dynamic Logic in PVS*, Technical Report SRI-CSL-02-04, Computer Science Laboratory, SRI International, 2002.

[12] D. Luckham, J. Kenney, L. Augustin, J. Vera, D. Bryan and W. Mann, *Specification and Analysis of System Architecture Using Rapide*, IEEE Transactions on Software Engineering, Special Issue on Software Architecture, 21(4), 1995.

[13] J. Magee, N. Dulay, S. Eisenbach and J. Kramer, *Specifying Distributed Software Architectures*, in Proceedings of the 5th European Software Engineering Conference (ESEC '95), Sitges, Spain, Lecture Notes in Computer Science, Springer-Verlag, 1995.

[14] N. Medvidovic, *ADLs and dynamic architecture changes*, in Proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints '96) on SIGSOFT '96 workshops, San Francisco, CA, 1996.

[15] N. Medvidovic, P. Oreizy, J. Robbins and R. Taylor, *Using Object-Oriented Typing to Support Architectural Design in the C2 Style*, in Proceedings of the ACM SIGSOFT '96 Fourth Symposium on the Foundations of Software Engineering, ACM SIGSOFT, San Francisco, CA, 1996.

[16] J. Spivey, *The Z Notation: A Reference Manual*, 2nd edition, Prentice-Hall International, 1992.

[17] M. Wermelinger, A. Lopes and J. Fiadeiro, *A Graph Based Architectural (Re)configuration Language*, in ESEC/FSE'01, V.Gruhn (ed), ACM Press, 2001.

Métrica para la Valoración de las Capacidades Humanas en el Proceso Software

Silvia T. Acuña
Universidad Autónoma de Madrid, Escuela Politécnica Superior
Madrid, España, 28049
silvia.acunna@uam.es

Saritha G. Figueroa
Universidad Nacional de Santiago del Estero, Departamento de Informática
Santiago del Estero, Argentina, 4200
sarithaf@unse.edu.ar

Ricardo D. Cordero
Universidad Nacional de Santiago del Estero, Departamento de Matemática
Santiago del Estero, Argentina, 4200
rcordero@unse.edu.ar

Resumen

Este artículo se basa en la premisa de que las capacidades o competencias conductuales de las personas influye en la eficacia y eficiencia del desempeño de un rol determinado en el proceso software. Se ha desarrollado un Modelo del Proceso Software Orientado a las Capacidades que define además de los elementos tradicionales del proceso (actividades, productos, técnicas, personas y roles): las capacidades humanas y las capacidades de los roles. La inclusión de las competencias conductuales en el modelo de proceso implica la definición de las relaciones capacidad-persona y capacidad-rol involucradas en el desarrollo de software. Estas relaciones son necesarias para asignar personas a diferentes actividades del proceso software. Se propone una métrica para el Procedimiento de Asignación de Personas a Roles. Esta métrica permite realizar una valoración y ponderación cuantitativa de la relación capacidad-rol. Además se ha realizado una validación empírica de la métrica ponderada propuesta y se presenta un caso de estudio que muestra la importancia de la valoración de atributos personales por medio de la aplicación de una medida con fundamento estadístico.
Palabras clave: capacidades humanas, roles, métrica de ajuste persona-rol, equipo de desarrollo, proceso software.

Abstract

This paper is based on the premise that people's capabilities or behavioural competencies influence the effectiveness and efficiency with which they perform a given role in the software process. A capabilities-oriented process model has been developed that, apart from traditional software process elements (activities, products, techniques, people and roles), defines human capabilities and role-related capabilities. The inclusion of behavioural competencies in the process model implies the definition of the capability/people and capability/role relationships involved in software development. These relationships are necessary for assigning people to different software process activities. We propose a metric for the Assignation of People to Roles Procedure. This metric can be used to assess and quantitatively weight the capability/role relationship. In addition, the proposed weighed metric is evaluated empirically, and a case of study is presented that shows the importance of assessing personal attributes by applying a statistics-based measure.
Key words: human capabilities, role, people/role fit metric, development team, software process.

1. Introducción

La producción de software es una actividad básicamente intelectual y social [22], por lo tanto el comportamiento no determinístico y subjetivo de las personas que participan en ella influye en forma decisiva en los resultados. La carencia

de especificación de las personas en el proceso software y de procedimientos definidos para asignarlas a roles según las capacidades que posean y las capacidades requeridas por cada rol implica el riesgo de que las personas ejecuten actividades no adecuadas a sus capacidades. Los modelos de proceso existentes no formalizan las habilidades y capacidades de cada miembro: gestores, desarrolladores, personal de soporte, profesionales, expertos, clientes y usuarios, para que el equipo del proyecto sea eficaz y eficiente [14][21].

Estos problemas ponen de manifiesto la necesidad de modelos integrales del proceso de construcción de software, que incorporen tanto las partes técnicas como humanas del proceso. En este contexto, se ha desarrollado un Modelo del Proceso Software Orientado a las Capacidades. Este modelo formaliza las capacidades o competencias conductuales de las personas del proceso software [2], es decir de quiénes llevan a cabo las actividades de cada uno de los procesos del modelo propuesto. En [3] se comprobó que la incorporación a los cinco elementos típicos de los modelos de proceso software (actividades, productos, técnicas, personas, roles) de las capacidades de las personas y de las capacidades de los roles, mejora el proceso de desarrollo de software. Sin embargo, la métrica incorporada en el procedimiento de asignación de personas a roles sólo considera una relación porcentual que mide coincidencias entre las capacidades requeridas por un rol y las que posee una persona determinada. De esta manera no se valorizan atributos ausentes o superadores.

El propósito de este trabajo es desarrollar una métrica que pondera atributos presentes, ausentes o superadores mediante la aplicación de una medida estadística para asignar personas a roles en el proceso software. Se realizó una validación empírica de este aporte, que permitió apreciar la importancia de utilizar una métrica ponderada en procesos básicamente sociales, como el proceso de construcción de software.

Este artículo está organizado de la siguiente forma. En la sección 2 se discuten los trabajos relacionados. La sección 3 presenta una visión global del Modelo del Proceso Software Orientado a las Capacidades y la definición de las relaciones capacidad-persona y capacidad-rol de dicho modelo. En las secciones 4 y 5 se describen los dos procedimientos donde intervienen las relaciones anteriores: el Procedimiento de Evaluación de Capacidades de las Personas y el Procedimiento de Asignación de Personas a Roles. En la sección 6 se propone una métrica para el Procedimiento de Asignación de Personas a Roles, aporte original de este trabajo. Esta métrica se ha validado con miembros de una organización española desarrolladora de software. En la sección 7 se presenta un caso de estudio. Por último, en la sección 8 se comentan las conclusiones.

2. Trabajos Relacionados

La necesidad de contar con personal altamente preparado para el desarrollo del software ha sido motivo de discusión desde los años 80 [5][7][23]. En la actualidad, en el área de la evaluación del proceso software existen: a) el People-CMM [6], focalizado en el factor recurso humano; b) el Proceso Software Personal [12], focalizado en el rendimiento individual; y c) el Proceso Software del Equipo [13], que trata la mejora del proceso software a nivel de equipo. Sin embargo, en el área de la modelización del proceso software, se observa una falta de conceptualización y formalización de la incorporación de las personas y de la interacción en la que participan [8][20][10][30].

El proceso de asignación de personas a roles es uno de los temas menos tratados en el desarrollo de software a pesar de su incuestionable importancia, prueba de esto es, por ejemplo, que constituye parte de las áreas de proceso de nivel 2 y 3 de People-CMM [6]. Generalmente en las organizaciones desarrolladoras de software, las personas se asignan a roles no sistemáticamente y de manera intuitiva [24]. Cada gestor del equipo que conoce a su gente realiza un procedimiento que puede ser sistemático pero no se basa en reglas pre-establecidas sino utiliza sus intuiciones y mecanismos propios. Dentro de una organización o bien en distintas organizaciones, si este procedimiento lo realizan varias personas cada una utiliza su procedimiento y cada una lo hace de una forma distinta, no se sigue un procedimiento formalizado, siendo este procedimiento no sistemático. Tales procedimientos de asignación no formalizan las habilidades y capacidades de los miembros del equipo de software para asegurar la identificación de las personas más adecuadas para ciertos roles.

Se han desarrollado diversos trabajos para sistemas de localización de expertos que ayudan a descubrir especialidades y competencias de los individuos dentro de la organización, mediante la exploración de evidencias implícitas de conocimiento. Por ejemplo CompMAP, es una herramienta para el mapeamiento de competencias en equipos de proyectos [29] que ofrece una forma de representación visual de perfiles de competencias y sus relaciones. ProPer [28] es un sistema que representa el perfil de una persona con un vector que contiene valores asociados a diferentes habilidades, e implementa dos métodos de evaluación de perfiles, uno compensatorio y otro no compensatorio. Sin embargo, estas herramientas no caracterizan los perfiles de roles involucrados en un desarrollo para asignar personas a roles según las capacidades que poseen las personas y las capacidades requeridas por cada rol a fin de asistir a los gestores de los equipos de desarrollo de software en esta actividad de asignación.

3. Modelo del Proceso Software Orientado a las Capacidades

La descripción completa del Modelo del Proceso Software Orientado a las Capacidades se puede encontrar en [2]. Este modelo ha sido formalizado en UML e implementado en una herramienta que se encuentra disponible en http://www.ls.fi.upm.es/spt/. No obstante, en este artículo nos centraremos en lo concerniente a las personas y sus capacidades.

Las relaciones *Capacidad-Rol* y *Capacidad-Persona* se han formalizado sobre la base de herramientas técnico-conceptuales de la ciencia del comportamiento organizacional y de la gestión integrada de personas [11][17][18], que, a su vez, se fundamentan en principios bien establecidos de la psicología, en general, y de la psicología laboral, en particular.

La relación Capacidad-Persona interviene en el Procedimiento de Evaluación de Capacidades de las Personas y la relación Capacidad-Rol en el Procedimiento de Asignación de Personas a Roles. Por lo tanto, para cada subproceso cultural propuesto, se define un procedimiento para ayudar en la construcción de modelos de las personas involucradas en los procesos software y de la asignación de roles en los proyectos de software.

En [3] se han estudiado y analizado listas de capacidades o competencias conductuales de las personas que se utilizan en la selección de candidatos para puestos de trabajo [18] validadas en el marco del Assessment Centre Method. Estas capacidades se han organizado en cuatro categorías: habilidades intrapersonales; habilidades organizativas; habilidades interpersonales; y habilidades directivas. Las definiciones de las cuatro categorías y de las capacidades asociadas a cada categoría se muestran en la tabla 1.

	CATEGORÍAS			
	Habilidades Intrapersonales	**Habilidades Organizativas**	**Habilidades Interpersonales**	**Habilidades Directivas**
DESCRIPCIÓN	Se trata de competencias conductuales de tipo elemental, básicas en el individuo, de cuyo desarrollo se ocupan los procesos de inculturación básica y de formación, y que resultan preparatorias de un desempeño eficaz para el desarrollo profesional.	Se trata de competencias conductuales relacionadas con el desempeño eficaz de una posición desde el punto de vista tanto de la actuación personal, individual como de la adaptación del profesional a la vida de una organización estructurada para evolucionar dentro de tal organización.	Se trata de competencias conductuales que resultan relacionadas con el éxito en las tareas que suponen contacto interpersonal para el correcto desempeño de las actividades del proceso. Este tipo de habilidades está íntimamente implicado con la eficacia en posiciones de contacto social.	Se trata de competencias conductuales que resultan imprescindibles para dirigir a otras personas dentro de la organización, orientando su desempeño, en diferentes niveles de supervisión y con distintos grados de responsabilidad.
CAPACIDADES	- Análisis - Decisión - Independencia - Innovación/creatividad - Juicio - Tenacidad - Tolerancia al estrés	- Auto-organización - Gestión del riesgo - Conocimiento del entorno - Disciplina - Orientación ambiental	- Atención al cliente - Capacidad de negociación - Empatía - Sociabilidad - Trabajo en equipo/cooperación	- Evaluación de los colaboradores - Liderazgo de grupos - Planificación y organización

Tabla 1: Clasificación de las capacidades

Las capacidades de la tabla 1 caracterizan la relación Capacidad-Persona. La definición de esta relación se utiliza en el Procedimiento de Evaluación de Capacidades de las Personas para la realización del Proceso de Evaluación de Capacidades, que consiste en la determinación de las capacidades que posee una persona a fin de posteriormente asignarle un rol en el proceso. El Proceso de Evaluación de Capacidades está íntimamente relacionado con el Proceso de Asignación de Personas a Roles, a través básicamente del elemento, capacidades. Las capacidades constituyen el elemento integrador entre el perfil de cada persona y el perfil de cada rol. Así, para realizar una asignación de personas a roles con posibilidades de éxito no se puede pronosticar sobre cómo se comportarán las personas, aunque se conozca las capacidades que poseen, sin conocer de la forma más exacta posible cuáles son las competencias conductuales que exige el rol. En consecuencia, hemos definido la relación Capacidad-Rol y se proporciona al jefe del proyecto de software un procedimiento que le facilite la realización de la tarea de asignación de personas a roles para crear la estructura del equipo del proyecto.

Por tanto, las capacidades se utilizan tanto en el Procedimiento de Evaluación de Capacidades de las Personas como en el Procedimiento de Asignación de Personas a Roles, y son determinadas para cada persona involucrada en el desarrollo de software y, asimismo, son asignadas a cada uno de los subprocesos del modelo de proceso propuesto.

3.1. Relación capacidad-persona

Para determinar qué factores de la personalidad o conductas indicadoras de la personalidad están asociadas con las personas involucradas en los procesos de desarrollo de software, proponemos la Tabla de Correspondencia, que se muestra en la tabla 2, entre cada una de las 20 capacidades (tabla 1) y los factores de la personalidad de un test psicométrico. En concreto, se utiliza el Test de Personalidad de tipo proyectivo: 16 PF-5, descrito por Russell y Karol [19]. Este test ha sido seleccionado, ya que es uno de los instrumentos más utilizados y por su adaptación y correlación con determinados test de personalidad para los psicólogos. El 16 PF-5 mide 16 rasgos primarios de personalidad identificados por Cattell et al. [4] tales como afabilidad, razonamiento y estabilidad, entre otros, que describen conductas humanas de primer orden, y cinco dimensiones globales de la personalidad tales como extraversión, ansiedad, dureza, independencia y auto-control. Cada uno de estos factores tiene un polo bajo (representado con el signo "-") y un polo alto (representado con el signo "+"). Por ejemplo, Estabilidad - describe una persona reactiva y emocionalmente cambiable y Estabilidad + una persona emocionalmente estable, adaptada y madura. Este test psicométrico permite la determinación de conductas indicadoras de las capacidades de la persona mediante el análisis factorial del conjunto de descriptores de la personalidad total.

Hemos elaborado la Tabla de Correspondencia (tabla 2) entre los factores de personalidad del test 16 PF-5 y las capacidades de las personas, a fin de evaluar la estructura de la personalidad mediante factores validados adecuadamente y correlacionados unos con otros, los del test 16 PF-5, y de disponer de una interfaz amigable con el usuario del proceso a través de las capacidades propuestas en este trabajo. Por ejemplo, para la fila de análisis (tabla 2), la conducta de la persona indica que es una persona con elevada potencia mental, perspicaz y de rápido aprendizaje (Razonamiento+), y a su vez denota una persona con sentido práctico y con los pies en la tierra, es decir realista y práctica (Abstracción-). Esta Tabla de Correspondencia se utiliza en el Procedimiento de Evaluación de Capacidades de las Personas para, una vez detectados los rasgos primarios y dimensiones globales de la personalidad del individuo a través del test 16 PF-5, integrarlos y determinar las capacidades pertinentes al desarrollo de software, como se describirá en la sección 7.

Para la definición de esta correspondencia se han realizado dos tareas: a) un análisis bidireccional tanto de los requisitos de personalidad de cada capacidad como de las facetas conductuales de cada factor de la personalidad; y b) una síntesis valorativa con la participación de psicólogos laborales de la Universidad Complutense de Madrid y de la empresa TEA, España.

Capacidades		Escalas y dimensiones del Test PF-5																
		Afabilidad	Razonamiento	Estabilidad	Dominancia	Atención a normas	Sansibilidad	Vigilancia	Abstracción	Privacidad	Aprensión	Apertura al cambio	Autosuficiencia	Perfeccionismo	Tensión	Ansiedad	Dureza	Independencia
Habilidades intrapersonales	Análisis		+						-									
	Decisión																+	
	Independencia				+													+
	Innovación / Creatividad								+			+						
	Juicio		+						-									
	Tenacidad			+										+				
	Tolerancia al estrés														-	-		
Habilidades organizativas	Auto-organización													+				
	Gestión del riesgo		+															
	Conocimiento del Entorno							+				+						
	Disciplina					+												
	Orientación ambiental						+											
Habilidades interpersonales	Atención al cliente	+					+											
	Capacidad de negociación	+								+								
	Empatía	+																
	Sociabilidad	+																
	Trabajo en equipo/cooperación												+					
Habilidades directivas	Evaluación de los colaboradores									+								
	Liderazgo de gruposPlanificación y organización				+						-		+					
	Planificación y organización														+			

Tabla 2: Correspondencia entre los factores de personalidad del test 16 PF-5 y las capacidades propuestas

3.2. Relación capacidad-rol

Proponemos la asignación de capacidades a roles que muestra la tabla 3. Para cada rol del proceso se han definido las capacidades necesarias para la consecución con éxito de las actividades asociadas. Las capacidades propuestas son las obligatorias para cada rol. Esto no implica que la persona pueda poseer otras capacidades deseables. Cada una de estas

capacidades se pondera en dos niveles: alto (A) y medio (M). Las ponderaciones indican un grado de requerimiento alto o bien medio de la capacidad.

Esta tabla ha sido elaborada a través de un análisis de las competencias conductuales requeridas para el desempeño eficaz de cada rol. Para ello, se ha llevado a cabo un proceso de reflexión que ha involucrado múltiples tareas de análisis: a) análisis de las actividades realizadas por cada rol; b) análisis de las situaciones críticas para el éxito en el desempeño de cada rol, clasificadas según las cuatro categorías de capacidades descritas anteriormente, considerando así situaciones individuales, organizativas, grupales o directivas; c) análisis de las 20 competencias conductuales propuestas, determinando las competencias que sean requeridas por cada una de dichas situaciones críticas para alcanzar un resultado positivo, pensando en todo momento que se trata de situaciones críticas sin cuya realización adecuada es imposible o muy improbable alcanzar el resultado deseado. De igual modo, se ha considerado que al hablar de competencias conductuales requeridas nos referimos a aquellas que son realmente imprescindibles (y no sólo deseables o, incluso, importantes). Se trata de competencias conductuales que debe poseer la persona que desempeña el rol y en cuya ausencia la situación crítica no puede realizarse completa o adecuadamente y, en consecuencia, el objetivo del subproceso correspondiente no se alcanza. El resultado de esta reflexión se ha reflejado en tablas similares a la tabla 3 por situación crítica y por rol. Finalmente, mediante sesiones participativas se ha repasado en forma sistemática y completa la lista de competencias conductuales indicando aquellas que resultan necesarias para resolver con éxito cada una de las situaciones críticas y la relevancia o importancia (alta o media) de cada una de ellas. En estas sesiones participativas han intervenido personas que realizan el rol que se está analizando, personas que desempeñan roles que le suministran productos de entrada a este rol y personas que consumen los productos generados por el rol en cuestión y la psicóloga laboral Marta Aparicio de la Universidad Complutense de Madrid, España, para la revisión de las capacidades consensuadas por los expertos en el desarrollo de software. Se ha determinado así, la Tabla Roles-Capacidades (tabla 3) propuesta en este trabajo. La justificación de las capacidades o habilidades que se necesitan para realizar adecuadamente cada rol considerado se detalla en [2].

PROCESOS	ROLES	CAPACIDADES																						
		Habilidades Intrapersonales								Habilidades Organizativas					Habilidades Interpersonales					Habilidades Directivas		
		Análisis	Decisión	Independencia	Innovación/creatividad	Juicio	Tenacidad	Tolerancia al estrés	Auto-organización	Gestión del riesgo	Conocimiento del entorno	Disciplina	Orientación ambiental	Atención al cliente	Capacidad de negociación	Empatía	Sociabilidad	Trabajo en equipo/cooperación	Evaluación de los colaboradores	Liderazgo de grupos	Planificación y organización
PROCESO DE SELECCIÓN DEL MCVS	Seleccionador del MCVS	A	A	M	M		M	M			M	M	M			M		M			
PROCESOS DE GESTIÓN DEL PROYECTO																					
Proceso de Iniciación, Planificación y Estimación del Proyecto	Planificador	A	A	M	M	A		A	M	A			M		A	M		A	M	A	A
Proceso de Seguimiento y Control del Proyecto	Controlador	A	A	M	M	A		A	M	A			M		A	M		A	M	A	A
Proceso de Gestión de Calidad del Software	Ingeniero de Calidad	A	A	M	M			M	M				M			M		A	M	A	A
Proceso de Asignación de Personas a Roles	Asignador de Roles	A	A	M	M			M	M				M			M		A	M	A	A
PROCESOS DE MODELIZACIÓN DEL SOFTWARE																					
PROCESOS DE EXPLORACIÓN																					
Proceso de Estudio del Dominio	Analista del Dominio Organizacional	A	M	M	A	M		M			A				M	M	M	A			
Proceso de Estudio de Viabilidad	Analista de Viabilidad	A	M	M	A	M		M			A				M	M	M	A			
PROCESOS DE CONCEPTUALIZACIÓN																					
Proceso de Análisis de Sistemas	Analista de Sistemas	A				A		M					M	A		M	M	A			
Proceso de Análisis del Conocimiento	Analista del Conocimiento	A				A		M					M	A		M	M	A			
Proceso de Análisis de Requisitos	Especificador de Requisitos	A				A		M					M	A		M	M	A			
PROCESOS DE FORMALIZACIÓN																					
Proceso de Diseño	Diseñador	A	A	M			M	M	A			M	M			M		M			
Proceso de Implementación e Integración	Implementador	A	A	M			M	M	A			M	M			M		M			
PROCESOS DE UTILIZACIÓN																					
Proceso de Instalación y Aceptación	Instalador						A	A	A			M	M	A		A		M			
Proceso de Operación y Soporte	Operador del Sistema						A	A	A			M	M	A		A		M			
Proceso de Mantenimiento	Mantenedor						A	A	A			M	M	A		A		M			
Proceso de Retiro	Gestor de Retiro						A	A	A			M	M	A		A		M			
PROCESOS DE SOPORTE DEL PROYECTO																					
PROCESOS DE PROTECCIÓN DE LA CALIDAD																					
Proceso de Verificación y Validación	Validador			M		M		M	A			A	M			M		A			
Proceso de Configuración	Gestor de Configuración			M		M		M	A			A	M			M		A			
PROCESOS DE TRANSFERENCIA TECNOLÓGICA																					
Proceso de Documentación	Documentalista							M				M	A			A		A			
Proceso de Formación	Formador							M				M	A			A		A			
PROCESOS DE COMUNICACIÓN																					
Proceso de Instrumentación del Equipo	Gestor del Equipo	A	M		M			M			M		M	M	M	A	A	A	A	A	A
Proceso de Adquisición de Información y Conocimientos	Elicitador	A	M		M			M			M		M	M	M	A	A	A	A	A	A

Tabla 3: Capacidades y rol por subproceso del modelo

4. Valorización de la Tabla Roles-Capacidades

En la relación Capacidad – Rol se establecen ponderaciones cualitativas en dos niveles, es decir cada capacidad representa una variable cualitativa que puede asumir un valor Alto o Medio. Para desarrollar la métrica que se propone en este trabajo se genera a partir de ésta una variable cuantitativa mediante la función de transformación que sigue:

$$X = f(x) = \begin{cases} 0.7 & si \quad x = Alto \\ 0.3 & si \quad x = Medio \\ 0 & si \quad x = No\ Requerido \end{cases} \tag{1}$$

Esta nueva variable (1) asume valores numéricos comprendidos entre cero (0) y uno (1). Sin embargo no todas las capacidades son igualmente importantes para desempeñar cada rol, por lo cual se define un método compensatorio que asigna pesos a cada una de las 20 capacidades (2).

El peso de la capacidad c_i se define como:

$$c_i = \frac{f(X_i = 0.7)*(0.7) + f(X_i = 0.3)*(0.3)}{Total\ de\ Roles} \tag{2}$$

En función de los valores ideales de cada capacidad y del peso asignado a la misma se puede determinar una "*media ponderada ideal*" (3) para cada rol [25].

$$\overline{XI}(R_K) = \frac{\sum (c_i * X_i)}{\sum c_i} \tag{3}$$

Donde:

$\overline{XI}_{(R_K)}$, representa la "*media ponderada ideal*" para el rol "k" (k = 1…22)

c_i, representa el peso de la capacidad "i" (i = 1 …20)

X_i, representa el valor de la capacidad "i" (i = 1 …20)

Se obtiene de esta manera una Tabla de Roles – Capacidades valorizada (tabla 4).

Roles	Habilidades intrapersonales							Habilidades organizativas					Habilidades interpersonales					Habilidades directivas			MEDIA PONDERADA IDEAL PARA CADA ROL
	Análisis	Decisión	Independencia	Innovación / Creatividad	Juicio	Tenacidad	Tolerancia al estrés	Auto-organización	Gestión del riesgo	Conocimiento del Entorno	Disciplina	Orientación ambiental	Atención al cliente	Capacidad de negociación	Empatía	Sociabilidad	Trabajo en equipo/cooperación	Evaluación de los colaboradores	Liderazgo de grupos	Planificación y organización	
Seleccionador del MCVS	0.7	0.7	0.3	0.3	0	0.3	0.3	0	0	0.3	0.3	0.3	0	0	0.3	0	0.3	0	0	0	0.23
Planificador	0.7	0.7	0.3	0.3	0.7	0	0.7	0.3	0.7	0	0	0.3	0	0.7	0.3	0	0.7	0.3	0.7	0.7	0.41
Controlador	0.7	0.7	0.3	0.3	0.7	0	0.7	0.3	0.7	0	0	0.3	0	0.7	0.3	0	0.7	0.3	0.7	0.7	0.41
Ingeniero de Calidad	0.7	0.7	0.3	0.3	0	0	0.3	0.3	0	0	0	0.3	0	0	0.3	0	0.7	0.3	0.7	0.7	0.33
Asignador de roles	0.7	0.7	0.3	0.3	0	0	0.3	0.3	0	0	0	0.3	0	0	0.3	0	0.7	0.3	0.7	0.7	0.33
Analista del Dominio organizacional	0.7	0.3	0.3	0.7	0.3	0	0.3	0	0	0.7	0	0	0	0.3	0.3	0.3	0.7	0	0	0	0.26
Analista de Viabilidad	0.7	0.3	0.3	0.7	0.3	0	0.3	0	0	0.7	0	0	0	0.3	0.3	0.3	0.7	0	0	0	0.26
Analista de Sistemas	0.7	0	0	0	0.7	0	0.3	0	0	0	0	0.3	0.7	0	0.3	0.3	0.7	0	0	0	0.27
Analista del Conocimiento	0.7	0	0	0	0.7	0	0.3	0	0	0	0	0.3	0.7	0	0.3	0.3	0.7	0	0	0	0.27
Especificador de requisitos	0.7	0	0	0	0.7	0	0.3	0	0	0	0	0.3	0.7	0	0.3	0.3	0.7	0	0	0	0.27
Diseñador	0.7	0.7	0.3	0	0	0.3	0.3	0.7	0	0	0.3	0.3	0	0	0.3	0	0.3	0	0	0	0.26
Implementador	0.7	0.7	0.3	0	0	0.3	0.3	0.7	0	0	0.3	0.3	0	0	0.3	0	0.3	0	0	0	0.26
Instalador	0	0	0	0	0	0.7	0.7	0.7	0	0	0.3	0.3	0.7	0	0.7	0	0.3	0	0	0	0.23
Operador del sistema	0	0	0	0	0	0.7	0.7	0.7	0	0	0.3	0.3	0.7	0	0.7	0	0.3	0	0	0	0.23
Mantenedor	0	0	0	0	0	0.7	0.7	0.7	0	0	0.3	0.3	0.7	0	0.7	0	0.3	0	0	0	0.23
Gestor de retiro	0	0	0	0	0	0.7	0.7	0.7	0	0	0.3	0.3	0.7	0	0.7	0	0.3	0	0	0	0.23
Validador	0	0	0.3	0	0.3	0	0.3	0.7	0	0	0.7	0.3	0	0	0.3	0	0.7	0	0	0	0.22
Gestor de configuración	0	0	0.3	0	0.3	0	0.3	0.7	0	0	0.7	0.3	0	0	0.3	0	0.7	0	0	0	0.22
Documentalista	0	0	0	0	0	0	0.3	0	0	0	0.3	0.7	0	0	0.7	0	0.7	0	0	0	0.17
Formador	0	0	0	0	0	0	0.3	0	0	0	0.3	0.7	0	0	0.7	0	0.7	0	0	0	0.17
Gestor del equipo	0.7	0.3	0	0.3	0	0	0.3	0	0	0	0.3	0	0.3	0.3	0.7	0.7	0.7	0.7	0.7	0.7	0.34
Elicitador	0.7	0.3	0	0.3	0	0	0.3	0	0	0.3	0	0.3	0.3	0.3	0.7	0.7	0.7	0.7	0.7	0.7	0.34
PESO DE CADA CAPACIDAD	0.4	0.3	0.2	0.2	0.2	0.2	0.4	0.3	0.1	0.1	0.2	0.3	0.3	0.1	0.4	0.1	0.6	0.1	0.2	0.2	4.81

Tabla 4: Tabla de Roles – Capacidades valorizada

5. Determinación de las Capacidades de las Personas

El Procedimiento de Evaluación de Capacidades de las Personas se centra en cómo determinar las capacidades de las personas involucradas en el proceso software. Este procedimiento está descrito en [3]. La salida del Procedimiento de Evaluación de Capacidades de las Personas está formada principalmente por dos modelos (figura 1): el Modelo de Factores de Personalidad y el Modelo de Lista de Capacidades de las personas. El primero, Modelo de Factores de Personalidad, involucra un perfil gráfico de los 16 factores primarios de la personalidad y de las cinco dimensiones globales del test 16 PF-5.

Organización Española

Informe Factores de Personalidad

ED

Nombre del proceso que realiza: .....

Rol que desempeña: .....

Fecha de la evaluación cultural: *21/6/2000*..... N° de informe: 60

Nombre de los evaluadores: Silvia Teresita Acuña y Marta Evelia Aparicio

A. Datos personales

Edad: 43

Nacionalidad: Argentina

Lugar de origen: Santiago del Estero

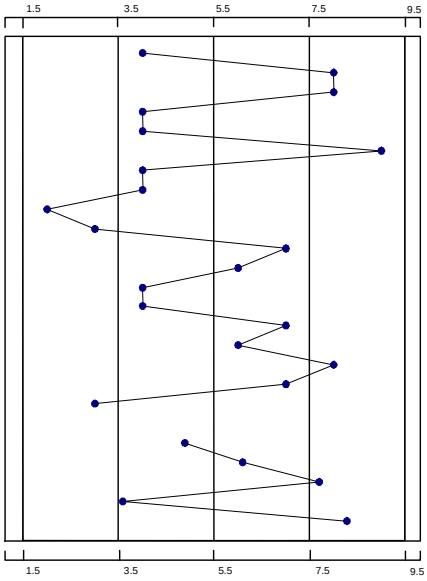
Sexo: ☐ Varón ☒ Mujer

Residencia habitual: .....

Estado civil: ☐ Soltero/a ☒ Casado/a ☐ Separado/a ☐ Divorciado/a ☐ Viudo/a

B. Perfil gráfico

		PD	DE
Afabilidad	A	13	4
Razonamiento	B	12	8
Estabilidad	C	19	8
Dominancia	E	10	4
Animación	F	10	4
Atención a las normas	G	21	9
Atrevimiento	H	8	4
Sensibilidad	I	9	4
Vigilancia	L	2	2
Abstracción	M	0	3
Privacidad	N	14	7
Aprensión	O	16	6
Apertura al cambio	Q1	11	4
Autosuficiencia	Q2	2	4
Perfeccionismo	Q3	18	7
Tensión	Q4	10	6
Manipulación de la imagen	MI	22	8
Infrecuencia	IN	1	7
Aquiescencia	AQ	37	3
EXTRAVERSIÓN	Ext		4,9
ANSIEDAD	Ans		6,1
DUREZA	Dur		7,7
INDEPENDENCIA	Ind		3,6
AUTO-CONTROL	AuC		8,3



Observaciones: .....

.....

C. Informe analítico sobre la persona

Efectúe una descripción redaccional con el detalle de apreciaciones sobre la persona evaluada, siguiendo cada factor de personalidad y dimensión global del test 16 PF-5 así como los puntos débiles y fuertes de los factores característicos de la persona detectados, de tal manera que pueda ser comprendida por cualquier profesional no especialmente familiarizado con los métodos y la terminología de evaluación psicológica.

Características de las conductas indicadoras de la personalidad:

..... Ver Volumen de Demostración de Desarrollos.....

Figura 1: Modelo de Factores de Personalidad para E.D

Organización Española	Informe Lista de Capacidades
	ED Nombre del proceso que realiza: ..... Rol que desempeña: ..... Fecha de la evaluación cultural: *28/7/2000*..... N° de lista: 60 Nombre de los evaluadores: Silvia Teresita Acuña y Marta Evelia Aparicio

A. Datos personales

Edad: 43 Nacionalidad: Argentina ☐ Lugar de origen: Santiago del Estero	Sexo: Varón ☒ Mujer Residencia habitual: .....

Estado civil:
Soltero/a ☒ Casado/a ☐ Separado/a ☐ Divorciado/a ☐ Viudo/a ☐

B. Capacidades o competencias conductuales

N°	Competencia	Nivel de competencia
1	Análisis	Alto
2	Decisión	Alto
3	Juicio	Alto
4	Tenacidad	Alto
5	Auto-organización	Alto
6	Gestión del riesgo	Alto
7	Disciplina	Alto
8	Trabajo en equipo/cooperación	Alto
9	Evaluación de los colaboradores	Alto
10	Planificación y organización	Alto

Observaciones: .....

C. Resumen general sobre la persona

Efectúe una descripción redaccional con el resumen de apreciaciones integradas sobre la persona evaluada, siguiendo las características de la personalidad y habilidades de la persona detectadas, de tal manera que pueda ser comprendida por cualquier profesional no especialmente familiarizado con los métodos y la terminología de evaluación psicológica.

Características de la personalidad y habilidades de la persona:

Elena puede tener dificultades en trabajos que impliquen contacto interpersonal, por lo que trabajos adecuados para ella serían aquellos técnicos y de mantenimiento de sistemas, así como programadora o investigadora.

Puede destacar profesionalmente en trabajos que implique pensar y razonar, así como en tareas que conlleven creatividad, docencia o alta exigencia de recursos intelectuales.

Se encontrará más a gusto si trabaja cerca de personas dominantes que supervisen y organicen su trabajo. Si no es así, su indecisión para asumir responsabilidades puede desmotivarla y generarle frustración. En esta línea, tiene una gran capacidad para aceptar normas de sus superiores y puede desarrollarse profesionalmente en trabajos que requieran autodisciplina y precisión.

Su perfil podría ser adecuado para desempeñar tareas de mando y gestión en organizaciones cuyos niveles jerárquicos estén claramente delimitados y no exista mucho contacto con sus subordinados; no es un líder nato.

Es una persona detallista lo que es adecuado para trabajos que requieran exactitud.

Puede desenvolverse con éxito en situaciones donde sea preciso disponer de diplomacia en el trato con los demás, así como para venderse a sí misma.

Es una persona organizada y perfeccionista, pero su exceso de perfección puede suponer que a veces su trabajo se ralentice.

Figura 1: Modelo de Factores de Personalidad para E.D (Cont.)

El Modelo de Lista de Capacidades de las personas, integra las valoraciones analíticas anteriores de forma coherente y presenta las capacidades de las personas junto con su nivel de requerimiento, determinadas a través de la Tabla de

Correspondencia. Además, este documento involucra una valoración sintética de las características de la personalidad y habilidades de la persona.

Es conveniente aclarar que este procedimiento no se realiza para cada proyecto, sino la idea involucra lo siguiente: a) la existencia en la organización de una base de datos con las capacidades del personal; b) la revisión de esta base cada año por las variaciones de personalidad individuales; y c) la utilización de esta información por parte del jefe de proyecto cuando tenga que asignar personas a roles.

6. Métrica para la Asignación de Personas a Roles

El Procedimiento de Asignación de Personas a Roles permite realizar la asignación de personas para desempeñar roles, es decir para llevar a cabo las actividades, según las capacidades que poseen las personas y que requieren los roles. Para alcanzar su objetivo, este Procedimiento conlleva un proceso de estructuración de los perfiles obtenidos en cuatro actividades: a) comparación, b) evaluación, c) seguimiento y consolidación, y d) documentación. Estas actividades, sus tareas y sus documentos de entrada y de salida asociados se muestran en la figura 2.

En la actividad de comparar perfil personal-perfil del rol, se analiza cada perfil de persona con cada perfil de rol buscando el mayor grado de ajuste entre el perfil personal y el del rol.

En esta actividad se trabaja con el Modelo de Factores de Personalidad generado en el Procedimiento de Evaluación de Capacidades de las Personas, y el Perfil de Capacidades por Rol valorizado (tabla 4).

A partir del Modelo de Factores de Personalidad se construye una tabla similar a la tabla 4 de Roles- Capacidades valorizada en la cual se ubican los valores particulares obtenidos por una persona determinada, se adiciona una columna con la "*media ponderada existente*", para cada rol y para cada persona (tabla 5).

Roles	Análisis	Decisión	Independencia	Innovación / Creatividad	Juicio	Tenacidad	Tolerancia al estrés	Auto-organización	Gestión del riesgo	Conocimiento del Entorno	Disciplina	Orientación ambiental	Atención al cliente	Capacidad de negociación	Empatía	Sociabilidad	Trabajo en equipo/cooperación	Evaluación de los colaboradores	Liderazgo de grupos	Planificación y organización	MEDIA PONDERADA PARA E.D.	MEDIA PONDERADA IDEAL PARA CADA ROL
Seleccionador del MCVS	0.7	0.7	0	0	0	0.7	0	0	0	0	0.7	0	0	0	0	0	0.7	0	0	0	**0.24**	0.23
Planificador	0.7	0.7	0	0	0.7	0	0	0.7	0.7	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.32	0.41
Controlador	0.7	0.7	0	0	0.7	0	0	0.7	0.7	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.32	0.41
Ingeniero de Calidad	0.7	0.7	0	0	0	0	0	0.7	0	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.28	0.33
Asignador de roles	0.7	0.7	0	0	0	0	0	0.7	0	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.28	0.33
Analista del Dominio organizacional	0.7	0.7	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.22	0.26
Analista de Viabilidad	0.7	0.7	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.22	0.26
Analista de Sistemas	0.7	0	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.18	0.27
Analista del Conocimiento	0.7	0	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.18	0.27
Especificador de requisitos	0.7	0	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.18	0.27
Diseñador	0.7	0.7	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	**0.28**	0.26
Implementador	0.7	0.7	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	**0.28**	0.26
Instalador	0	0	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.18	0.23
Operador del sistema	0	0	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.18	0.23
Mantenedor	0	0	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.18	0.23
Gestor de retiro	0	0	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.18	0.23
Validador	0	0	0	0	0.7	0	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.19	0.22
Gestor de configuración	0	0	0	0	0.7	0	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.19	0.22
Documentalista	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.11	0.17
Formador	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.11	0.17
Gestor del equipo	0.7	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.23	0.34
Elicitador	0.7	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.23	0.34
PESO DE CADA CAPACIDAD	0.4	0.3	0.2	0.2	0.2	0.2	0.4	0.3	0.1	0.1	0.2	0.3	0.3	0.1	0.4	0.1	0.6	0.1	0.2	0.2	4.81	4.81

Tabla 5: Tabla de Roles – Capacidades valorizada para una persona determinada

La "*media ponderada existente*" (4) se calcula de la siguiente forma:

$$\overline{XE}(R_K) = \frac{\sum (c_i * X_i)}{\sum c_i} \qquad (4)$$

Donde:

$\overline{XE}_{(R_K)}$, representa la "*media ponderada existente*" para el rol "k" (k = 1…22) y para una persona determinada

c_i, representa el peso de la capacidad "i" (i = 1 …20)

X_i, representa el valor de la capacidad "i" (i = 1 …20) obtenido por una persona determinada

Sobre la base de estas dos entradas, se comparan las medias ponderadas (ideal y existente) para cada persona y cada rol. La regla de asignación considera:

Si la media ponderada existente es mayor o igual a la media ponderada ideal para cada rol ($\overline{XE}_{(R_K)} >= \overline{XI}_{(R_K)}$), entonces la persona se asigna al rol. En caso contrario ($\overline{XE}_{(R_K)} < \overline{XI}_{(R_K)}$), se determina la desviación estándar (D) y si se encuentra a una vez esta medida se puede también asignar al rol. Esto garantiza que la media ponderada existente se encuentra a una distancia cercana a la media ponderada ideal con un nivel de significancia del 16%. Si la media ponderada existente se encuentra a dos veces la desviación estándar se recomienda la inclusión de la persona en un programa de capacitación [25], [26], [27].

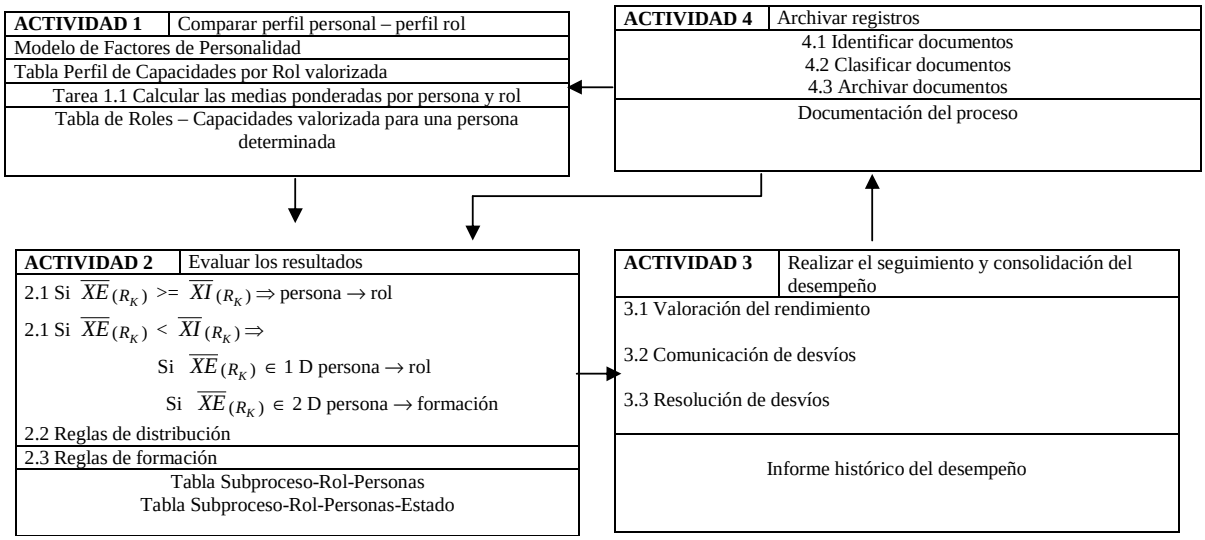


Figura 2: Procedimiento de Asignación de Personas a Roles

Además de las reglas de asignación y de formación, se han definido reglas de distribución. Por ejemplo, en un proyecto particular, ante la ausencia de personas con medias ponderadas que superen a la media ponderada ideal o se encuentren dentro de una vez el desvío estándar para los roles requeridos en dicho proyecto, se comparan las distancias a la media ponderada ideal de todas las personas consideradas para los roles involucrados y se seleccionan las personas con menor distancia a dicha media y con informes sintéticos más ajustados al perfil del rol. Así, se obtienen: la *Tabla Subproceso-Rol-Personas* y la *Tabla Subproceso-Rol-Personas-Estado*, donde Estado puede ser participación o formación.

Ambos modelos, ingresan en la actividad Realizar el seguimiento y consolidación del desempeño, donde se asegura que el efecto del ajuste rol-persona perdure, valorando si su asignación personalizada ha logrado mejorar el rendimiento actual, comunicando los desvíos y asegurando que no disminuya aquel efecto en virtud de la aplicación de técnicas probadas, que apoyan y sostienen la asignación realizada. Se obtiene el *Informe Histórico del Desempeño* que se documenta convenientemente en la actividad Archivar registros, y que se debe tener en cuenta en las nuevas asignaciones de roles que se realicen a través de este procedimiento.

7. Caso de Estudio

Para comprobar la relación personas-roles-capacidades y su procedimiento de asignación, se ha diseñado una validación empírica centrada en la aplicación de la métrica propuesta para el Procedimiento de Asignación de Personas a Roles. Esta validación se realizó en una organización española desarrolladora de software. Sus proyectos se han

desarrollado siguiendo las actividades del modelo de proceso aquí propuesto [2]. En este trabajo se describe la asignación de la persona E.D., cuyo Modelo de Factores de Personalidad se muestra en la figura 1.

Roles	Análisis	Decisión	Independencia	Innovación / Creatividad	Juicio	Tenacidad	Tolerancia al estrés	Auto-organización	Gestión del riesgo	Conocimiento del Entorno	Disciplina	Orientación ambiental	Atención al cliente	Capacidad de negociación	Empatía	Sociabilidad	Trabajo en equipo/cooperación	Evaluación de los colaboradores	Liderazgo de grupos	Planificación y organización	MEDIA PONDERADA PARA E.D.	MEDIA PONDERADA IDEAL PARA CADA ROL
Seleccionador del MCVS	0.7	0.7	0	0	0	0.7	0	0	0	0	0.7	0	0	0	0	0	0.7	0	0	0	**0.24**	0.23
Planificador	0.7	0.7	0	0	0.7	0	0	0.7	0.7	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.32	0.41
Controlador	0.7	0.7	0	0	0.7	0	0	0.7	0.7	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.32	0.41
Ingeniero de Calidad	0.7	0.7	0	0	0	0	0	0.7	0	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.28	0.33
Asignador de roles	0.7	0.7	0	0	0	0	0	0.7	0	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.28	0.33
Analista del Dominio organizacional	0.7	0.7	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.22	0.26
Analista de Viabilidad	0.7	0.7	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.22	0.26
Analista de Sistemas	0.7	0	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.18	0.27
Analista del Conocimiento	0.7	0	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.18	0.27
Especificador de requisitos	0.7	0	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0.18	0.27
Diseñador	0.7	0.7	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	**0.28**	0.26
Implementador	0.7	0.7	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	**0.28**	0.26
Instalador	0	0	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.18	0.23
Operador del sistema	0	0	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.18	0.23
Mantenedor	0	0	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.18	0.23
Gestor de retiro	0	0	0	0	0	0.7	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.18	0.23
Validador	0	0	0	0	0.7	0	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.19	0.22
Gestor de configuración	0	0	0	0	0.7	0	0	0.7	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.19	0.22
Documentalista	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.11	0.17
Formador	0	0	0	0	0	0	0	0	0	0	0.7	0	0	0	0	0	0.7	0	0	0	0.11	0.17
Gestor del equipo	0.7	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.23	0.34
Elicitador	0.7	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7	0.7	0	0.7	0.23	0.34
PESO DE CADA CAPACIDAD	0.4	0.3	0.2	0.2	0.2	0.2	0.4	0.3	0.1	0.1	0.2	0.3	0.3	0.1	0.4	0.1	0.6	0.1	0.2	0.2	4.81	4.81

Tabla 6: Tabla de Roles – Capacidades valorizada para la persona E.D.

En el caso de estudio puede observarse que E.D. obtiene valores de medias ponderadas existentes superiores a las ideales en tres roles, seleccionador del modelo de ciclo de vida del software (MCVS), diseñador e implementador respectivamente. Por lo cual la actividad 2 del Procedimiento de Asignación de Personas a Roles recomienda asignar a E.D. en estos roles. Sin embargo en la organización en la que se llevó a cabo la experiencia, no se contaba con personal para asumir el rol de planificador, por lo que se tomó la decisión de incluir a E.D. en dicho rol. Esta decisión se pudo tomar con fundamento desde la métrica, ya que la media ponderada existente se encuentra dentro de 1 vez el desvío estándar con respecto a la media ponderada ideal, lo cual garantiza un nivel de significancia del 16%, es decir que existe un 84% de confianza de que este valor es cercano o superador del valor de la media ponderada ideal para el rol de planificador.

8. Conclusiones

El comportamiento no determinístico y subjetivo de las personas que participan en el proceso software influye en los resultados del proceso de desarrollo de software. Por lo tanto sus capacidades deben incorporarse en el modelo de proceso para la asignación de personas a roles según las capacidades que aquéllas poseen y que requieren los roles. La incorporación de la métrica detallada en este trabajo permite asegurar la sistematización del procedimiento de asignación de personas a roles, mediante una medida estadística que pondera las capacidades que posee cada persona y las que requiere cada rol del proceso de construcción de software. La propuesta presentada en este artículo proporciona:

- Descripciones formalizadas y valorizadas de las capacidades tanto de las personas como de los roles y de sus interacciones en el proceso software, mejorando la eficacia y eficiencia del procedimiento de asignación de personas a roles.
- Una métrica para el Procedimiento de Asignación de Personas a Roles que contempla la importancia de cada capacidad para el desempeño de cada rol.

Concretamente, a partir de la experiencia realizada para validar la aplicación de la métrica en el Procedimiento de Asignación de Personas a Roles en una organización desarrolladora de software de modo similar al caso de estudio presentado, se comprobó que ésta permite mejorar la gestión de las personas en el desempeño de roles según sus capacidades, al realizar en forma eficaz y eficiente las actividades asignadas, proporcionando guías definidas y sistemáticas.

Referencias

[1] T. K. Abdel-Hamid y S. E. Madnick, *Software Project Dynamics: An Integrated Approach*. (Prentice Hall, Englewood Cliffs, 1991).

[2] S. T. Acuña, *Capacities-Oriented Software Process Integral Model*. Ph.D. tesis, Universidad Politécnica de Madrid (2002).

[3] S. T. Acuña y N. Juristo, Assigning People to Roles in Software Projects. Software: Practice & Experience. 34 : 675-696. (John Wiley & Sons, 2004).

[4] R. B. Cattell, A. K. Cattell y H. E. P. Cattell, *Sixteen Personality Factor Questionnaire*, Fifth Edition. (Institute for Personality and Ability Testing, 1993).

[5] J. Cougar y R. Zawacky, *Managing and Motivating Computer Personnel.* (Wiley, 1980).

[6] B. Curtis, W.E. Hefley and S.A. Miller. "People Capability Model (P-CMM)". Version 2.0. Software Engineering Institute. Maturity Model CMU/SEI – 2001-MM-001, 2001.

[7] T. De Marco y T. Lister, Peopleware. (Pittsburgh, P.A, 1994).

[8] G. Engels y L. Groenewegen, "SOCCA: Specifications of coordinated and cooperative activities". In Software Process Modelling and Technology, Eds. A. Finkelstein, J. Kramer y B. Nuseibeh, cap. 4. (Research Studies Press, 1994) 71-102.

[9] A. Finkelstein, J. Kramer y B. Nuseibeh, Software Process Modelling and Technology. (Research Studies Press, 1994).

[10] A. Fuggetta, "Software process: A roadmap". In The Future of Software Engineering, Ed. A. Finkelstein, (ACM Press, 2000) 27-34.

[11] G. Hamel y C. Prahalad, Competing for the Future. (Harvard Business School Press, 1994).

[12] W. S. Humphrey, Introduction to the Personal Software Process. SEI Series in Software Engineering. (Addison-Wesley, 1997).

[13] W. S. Humphrey, Three Dimensions of Process Improvement. Part III: The Team Software Process. (Crosstalk, 1998).

[14] W. S. Humphrey, Managing Technical People: Innovation, Teamwork and the Software Process. (Addison Wesley, 1998).

[15] I. Jacobson, G. Booch y J. Rumbaugh, The Unified Software Development Process. (Addison Wesley, 1999).

[16] N. Juristo y A. M. Moreno, Basics of Software Engineering Experimentation. (Kluwer Academic Publishers, 2001).

[17] E. Molleman y M. Broekhuis, "Sociotechnical systems: Towards an organizational learning approach". Journal of Engineering and Technology Management 18 (2001) 271-294.

[18] J. L. Moses y W. C. Byham, Applying the Assessment Center Method. (Pergamon, 1997).

[19] M. T. Russell y D. L. Karol, 16PF Fifth Edition Administrator's Manual. (Institute for Personality and Ability Testing, 1994).

[20] S.-Y. Min y D.-H. Bae, "MAM nets: A Petri-net based approach to software process modeling, analysis and management". *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering* (Junio 1997) 78-86.

[21] J. Slomp y E. Molleman, "Cross-training policies and team performance". *International Journal of Production Research* 40 (2002) 1193-1219.

[22] I. Sommerville y T. Rodden, "Human, social and organisational influences on the software process". Lancaster University, Computing Department, Cooperative Systems Engineering Group, Technical Report: CSEG/2/1995 (1995) 1-21.

[23] R. Whitaker, Managing Software Maniacs. (Wiley, 1994).

[24] R. van Solingen and E. Berghout. "From Process Improvement to People Improvement – Enabling Learning in Software Development". In: Project Control: The Software Factor (eds. K. Maxwell, R. Kusters, E. Van Veenendaal and A. Cowderoy), Shaker Publications, Maastricht, 2000.

[25] G. W. Snedecor, W. G. Cochran. "Métodos estadísticos". Compañía Editorial Continental, México 1979.

[26] A. Vegas Pérez. Estadística – Aplicaciones Econométricas y Actuariales. (Pirámide, 1981).

[27] E. Crivisqui. "Análisis Factorial de Correspondencias: Un instrumento de Investigación en Ciencias Sociales". Ed. del Laboratorio de Informática Social. Universidad Católica de Asunción. 2000.

[28] Y. Sure, A Maedche, S. Staab. "Leveraging Corporate Skill Knowledge – From Proper to OntoProper. Proc. of the 3rd Int. Conf. on Practical Aspects of Knowledge Management – PAKM2000, pp. 22/1-22/9, Basel, Switzerland, 2000.

[29] P. Waterson, S. Weibelzahl, D. Pfahl. "Software Process Modelling: Socio Technical Perspectives". In: Software Process Modelling (eds. S. T. Acuña y N. Juristo), Springer, NY, 2005.

"LIDIS": Consideraciones para la Generación, Maduración y Aplicación de conocimiento en Ingeniería de Software

Hugo F. Arboleda
Universidad de San Buenaventura, Facultad de Ingenierías.
Cali, Colombia,
huarbole@usb.edu.co

y
Jaime A. Chavarriaga
Universidad de San Buenaventura, Facultad de Ingenierías.
Cali, Colombia,
jaime@usb.edu.co

Abstract

The Software engineering retakes knowledge of diverse sources and approaches the development of software of quality at industrial level. Thus, it constructs knowledge around the practices of development of software. This knowledge takes shape in the definition of processes, methods and models that can be applied by the engineers in their professional activities. Part of the constructed knowledge as result of investigations in Software engineering is based on the revision and formalization of heuristics arisen from the real experience in the pursuit of processes for the construction of software products. In general, this knowledge is applicable in some contexts, doing necessary to validate it and to mature it for its application in different contexts. This can be obtained making use of a investigative strategy that includes phases of (1) investigation and initial development, (2) applied Investigation, and (3) Transference. Formalizing the process of investigation in software engineering it is managed to identify, to formalize and to theorize on the best practices in the area, and on their general applicability in different contexts. The investigation methods not necessarily must follow the classic directives of the quantitative experimental investigation.

Keywords: Engineering, Method of Investigation, Investigative Strategy, Software engineering, Contexts of development.

Resumen

La Ingeniería de Software retoma conocimientos de diversas fuentes y aborda el desarrollo de software de calidad a nivel industrial. Así, construye conocimiento en torno a las prácticas de desarrollo de software. Este conocimiento se concreta en la definición de procesos, métodos y modelos que pueden ser aplicados por los ingenieros en sus actividades profesionales. Parte del conocimiento construido como resultado de investigaciones en Ingeniería de Software se basa en la revisión y formalización de heurísticas surgidas de la experiencia real en el seguimiento de procesos para la construcción de productos de software. En general, este conocimiento es aplicable en algunos contextos, haciendo necesario validarlo y madurarlo para su aplicación en contextos diferentes. Esto se puede lograr haciendo uso de una estrategia investigativa que incluya fases de (1) investigación y desarrollo inicial, (2) Investigación aplicada, y (3) Transferencia. Formalizando el proceso de investigación en ingeniería de software se logra identificar, formalizar y teorizar sobre las mejores prácticas en el área, y sobre su aplicabilidad general en diferentes contextos. Los métodos de investigación no necesariamente deben seguir los lineamientos clásicos de investigación experimental cuantitativa.

Palabras Claves: Ingeniería, Método de Investigación, Estrategia Investigativa, Ingeniería de Software, Contextos de desarrollo.

1 INTRODUCCIÓN

La Ingeniería de Software es una disciplina relativamente joven, sus inicios se remontan a mediados de la década de los sesenta cuando algunos investigadores se propusieron la tarea de definir nuevas formas de construir software de manera más efectiva y confiable.

La forma como se construye el conocimiento en la Ingeniería de Software ha sido una cuestión importante, incluso desde sus comienzos. Los métodos de la Ingeniería de Software, como manifestaciones de ese conocimiento propio, han surgido de una gran variedad de fuentes y formas de trabajo, en algunas ocasiones como apropiaciones y adaptaciones de métodos existentes en otras disciplinas.

En los últimos años, a partir de algunos estudios sobre el desarrollo de las investigaciones de la Ingeniería de Software en el pasado [4][19] que han mostrado su relativamente bajo impacto en las prácticas profesionales, y la ausencia de métodos de investigación, se ha generado una polémica sobre las formas como las investigaciones deben desarrollarse en esta área. Nuevas perspectivas tendientes a la conformación de diferente métodos de investigación han surgido: Ingeniería de Software Experimental [4], Ingeniería de Software Empírica [16], y Métodos Cualitativos de Investigación en Ingeniería de Software [17], entre otros.

Un punto importante que debe considerarse al momento de definir la estrategia con la que la investigación puede desarrollarse, es la naturaleza misma de la Ingeniería de Software. Algunos autores conciben a la Ingeniería de Software como una división de la Ingeniería de Sistemas [6], otros como una división de las Ciencias de la Computación [15], y otros como una profesión y una ciencia a la vez [7]. Cada uno de estos autores propone diferentes esquemas de investigación acorde con su noción de lo que es la Ingeniería.

Para introducirse en el tema de la forma de generación de conocimiento, y el tipo de conocimiento generado, es importante revisar la naturaleza básica de las ingenierías en general, y de la ingeniería de software en particular. La segunda parte de este documento presenta una reflexión acerca del tema. La tercera parte presenta conceptos generales necesarios para generar conocimiento validado en ingeniería de software. La cuarta parte presenta una propuesta de estrategia de investigación, que siguiendo las consideraciones presentadas, incluye principios orientados a la relatividad de la validez del conocimiento generado, de acuerdo al contexto de aplicación. La quinta parte presenta resultados generales obtenidos del uso de la estrategia. Finalmente se presentan un conjunto de conclusiones.

2 Ingeniería: ¿Disciplina Científica o Tecnológica?

Las disciplinas científicas centran su campo de conocimiento en torno a la definición y comprensión de un objeto de estudio, típicamente proponiendo métodos y mecanismos tendientes a establecer teorías comprensivas alrededor de este conocimiento y a su verificación a través de mecanismos empíricos. Su propósito parece definirse más en términos de la comprensión y control de algunos fenómenos del mundo.

En las disciplinas de tipo tecnológico, por otro lado, su campo de conocimiento se da por la convergencia, debate y cooperación de diversas disciplinas en torno al desarrollo de una práctica propia. Su propósito parece definirse más en términos del desarrollo de una serie de actividades y de tareas en la vida laboral.

La Ingeniería busca la aplicación de métodos y principios de las diversas ciencias y disciplinas para la creación, construcción y mejoramiento de productos con adecuados niveles de calidad, economía y tiempo de desarrollo. La ingeniería tiene el propósito práctico de generar productos y soluciones reales a problemas y situaciones en el mundo real.

La naturaleza práctica de la ingeniería, la ha llevado a aplicar conocimientos y teorías formuladas por disciplinas científicas (que se centran en la descripción y comprensión de fenómenos), y a formular teorías y métodos, centrados en la aplicación práctica de los primeros. La idea de aplicación práctica de las ciencias, parece ser la concepción moderna sobre las ingenierías.

2.1 Aproximación General a la Ingeniería

La aproximación de ingeniería se centra en la utilización de modelos abstractos para el proceso de construcción de soluciones. Definir el proceso en función de modelos permite involucrar disciplinas como la Ingeniería de Software, en donde algunos métodos de desarrollo de productos no se centran en las “ciencias tradicionales”.

El proceso de generación de productos de software parte de la comprensión del problema en el mundo real. Para su comprensión, el ingeniero establece modelos abstractos que le permiten comprender y analizar el problema. Estos modelos pueden ser de diversa naturaleza: modelos matemáticos, modelos estadísticos, modelos químicos, modelos gráficos de los pasos de un proceso, entre otros.

En ingeniería se llama método al conjunto de modelos y transformaciones que permiten al ingeniero construir soluciones específicas a partir de determinado tipo de problemas. El aspecto científico del método se centra en la capacidad de los modelos de transformarse en otros de manera predecible y sistemática, además de su capacidad de predecir confiable y verazmente el comportamiento de las soluciones en el mundo real.

Los métodos obtenidos a nivel experimental, de laboratorio o en entornos académicos, deben ser adaptados en procesos que permiten la construcción masiva de productos. De esta forma por ejemplo, un método obtenido en laboratorio para sintetizar una proteína, producir determinado alimento o construir un artefacto de software, debe adaptarse en un proceso que permita la producción masiva de esos productos en una fábrica o industria.

Las actividades de mejoramiento continuo y de adaptaciones de los procesos de producción masiva, son también actividades naturales de los ingenieros. Actividades para las cuales, los ingenieros aplican métodos y modelos de igual forma como fueron generadas las primeras soluciones.

2.2 La Ingeniería como Área Tecnológica

Aunque en la vida cotidiana suele hacerse referencia a la tecnología solo como un conjunto de herramientas o dispositivos, el origen del término esta referido al estudio de las técnicas. Resulta más acertado definir la tecnología como un área de conocimientos, que parte de necesidades y problemas humanos, que se vale del diseño para el desarrollo de soluciones, y que se concreta en la producción de conocimiento alrededor de instrumentos, modelos y procedimientos.

Es posible clasificar el conocimiento generado en tres tipos: Conocimiento Científico, Conocimiento Técnico, y Conocimiento Tecnológico. El Conocimiento Científico, surge de las disciplinas científicas, típicamente brinda explicaciones sobre los fenómenos del mundo, incluyendo fenómenos naturales, sociales y culturales, entre otros. El Conocimiento Técnico, típicamente relacionado con los artesanos, se refiere al conocimiento necesario para el desarrollo de actividades concretas. Este se relaciona con la experiencia y el conocimiento empírico de varias generaciones en la construcción o desarrollo de prácticas específicas. Finalmente, el Conocimiento Tecnológico, surge del estudio y reflexión sobre el conocimiento técnico. Este plantea nuevas y mejores formas de desarrollar las prácticas tradicionales, empleando para esto todo el conocimiento científico y disciplinar a su alcance.

Basándose en la aplicación de conocimientos de diversas disciplinas, la Ingeniería desarrolla una variedad de nuevos conocimientos tecnológicos que representan teorías de tipo práctico en torno al desarrollo de los productos de los que se ocupa, los métodos para su construcción, y la forma como se desarrollan estos en la vida real.

2.3 La Ingeniería de Software como Profesión Tecnológica

Fritz Bauer estableció una de las primeras definiciones de ingeniería de software en la década de los 60: "*Ingeniería de Software es el establecimiento y uso de principios robustos de ingeniería, orientados a obtener software que sea fiable y funcione de manera eficiente sobre máquinas reales*". [14]

Sin embargo, a partir de ese momento, una gran variedad de nuevas definiciones surgieron, generando de fondo una dualidad sobre el término. Así, se han originado una variedad de nociones sobre lo que ella misma debería ser, sobre el tipo de conocimiento que debería abordar y sobre los métodos de investigación que debería usar.

Concebida como una profesión tecnológica, la Ingeniería de Software apropia el saber aportado por disciplinas como las Ciencias de la Computación, la Lógica, la Matemática, la Administración y las Ciencias Humanas, entre otras, para el análisis, diseño e implementación de productos de Software a nivel industrial. Las comunidades académicas y profesionales de la Ingeniería de Software no surgen solo a partir de un objeto disciplinar, sino también de las relaciones con usuarios concretos que demandan servicios específicos.

La Ingeniería de Software es una profesión de naturaleza tecnológica, que como tecnología y fenómeno cultural, involucra un campo de conocimiento sobre los procesos de construcción de software, los métodos de articulación de personas, procesos y recursos tecnológicos, en pos de la producción, aseguramiento de calidad y comercialización de productos de software, y sobre aquellos que involucran la creación y perfeccionamiento de herramientas y tecnofactos que permiten mejorar la eficiencia de esos procesos.

3 CONSTRUCCIÓN DE CONOCIMIENTO EN INGENIERÍA DE SOFTWARE

La Ingeniería de Software plantea sus conocimientos en torno al desarrollo de su práctica profesional y a la tecnología propia del desarrollo de software. Debido a su propia naturaleza, la construcción del conocimiento corresponde a un proceso de investigación tecnológica.

Los procesos de construcción de conocimiento en las ingenierías se soportan en una estructura cognitiva compleja que incluye asociaciones conceptuales de alto nivel en un determinado campo. De manera natural, el conocimiento en ingeniería de software se construye en busca de dar respuestas adecuadas a las necesidades y demandas del medio, a partir de considerar el contexto como elemento clave que alude a la pertinencia de las soluciones en relación con la problemática planteada.

La ingeniería en general se centra en la generación de los productos, y en la capacidad de generación de procesos de producción de los mismos. Estos procesos deben ser repetibles y rigurosos, deben permitir garantizar ciertos niveles de calidad y éxito, y no necesariamente son útiles en cualquier contexto. En Ingeniería de Software, estos procesos incluyen la ejecución de actividades identificadas como buenas prácticas de desarrollo, que de manera particular pueden incluir prácticas de diseño bajo paradigmas identificados de manera específica.

En algunos casos, el método que se aplica en Ingeniería para la construcción de conocimiento tecnológico, es la formalización de las experiencias, la identificación de las mejores prácticas, y la formalización de las heurísticas relacionadas con la construcción de los productos y servicios de los cuales se ocupa [11][12]. Es importante entonces considerar métodos de construcción de conocimiento que no solo se asemejen a procesos de investigación científica, sino mejor a procesos de investigación e innovación tecnológica, que se basan en la aplicación práctica de conceptos basados en la experiencia, que luego de validarse se establecen como parte del conocimiento generado.

Uno de los principales cuestionamientos a estas formas de investigación (basadas en heurísticas) se centran en las formas de validación de sus resultados. Diagnósticos realizados en el pasado por Basili, Taschi y Zelkowitz [4][19] han mostrado que gran parte del conocimiento propio de la Ingeniería de Software no parece ofrecer mecanismos confiables de validación. Pareciendo para algunos que muchos de ellos son meras especulaciones sobre unas pocas experiencias exitosas. Partiendo de esto se han propuesto métodos de investigación con gran énfasis en el componente experimental, en donde los componentes de verificación de las teorías son fundamentales para la aceptación de los resultados de las mismas.

Los métodos de investigación de ingeniería de software empírica, y los métodos cualitativos, han resultado adecuados en algunos casos para la Ingeniería de Software. Esto se debe en parte a que las empresas de desarrollo de software, entorno de trabajo de los proyectos de ingeniería de software, en general cuando alcanzan cierto nivel de madurez, están dispuestas a abordar estudios experimentales sobre los procesos que se llevan en su interior. Así, luego recopilan información empírica que es analizada y formalizada. De otro lado, las investigaciones que se llevan a cabo en condiciones controladas (laboratorios, universidades, etc.) conducen a métodos que no necesariamente aplican a todos los contextos. De hecho, algunos resultados nunca llegan a ser utilizados de manera total, sin que con esto deje de ser importante su aporte a la disciplina.

En general, las investigaciones pueden concentrarse en experiencias exitosas (o precisamente en las que no tienen éxito), con el fin de reconocer nuevas y mejores formas de trabajo, pero no puede suponer que son necesariamente válidas para todo tipo de contexto. Los aspectos concernientes a la validación son necesarios para lograr una mayor madurez de la profesión [6]. En términos generales, en la actualidad no existen modelos sobre los procesos de desarrollo de software que cuenten con un nivel de aspecto científico tal que permita hacer experimentación a partir de la simulación. El hecho de que los Ingenieros de Software deban trabajar con algoritmos, modelos cuantitativos y esquemas de ciencias de la computación, no es razón suficiente para creer que deban emplear los mismos métodos de investigación y validación que pueden usarse en esas áreas. No debe confundirse la Ingeniería de Software con las ciencias de la Computación [1][5][15].

3.1 Contextos de aplicación del conocimiento construido

La construcción de conocimiento no aplicable en todos los contextos se ve respaldada por la necesidad de adaptación de los métodos, a procesos dentro de la industria. El conocimiento generado debe ser adaptado a contextos propios de las organizaciones. Sin embargo, el que éste conocimiento no siempre sea aplicable en toda organización no indica que no

es válido. Se trata de una vista posmoderna, que admite el conocimiento generado como relativo al contexto de aplicación. El punto entonces está en poder definir y delimitar el contexto de validez del conocimiento construido.

Las diferentes tecnologías, teorías y métodos sufren un proceso de "maduración", y adaptación al contexto que solo puede lograrse a través de su aplicación en entornos empresariales reales. Algunos métodos de ingeniería de software, como por ejemplo el Rational Unified Process (RUP), ha evolucionado a través de diferentes versiones fruto de un proceso iterativo establecido desde su creación a mediados de los noventa. Sin embargo, los procesos de maduración no siempre logran la adaptación del conocimiento generado a un contexto particular. Por ejemplo, buenas prácticas exitosas en contextos de desarrollo en equipos grandes pueden no ser aplicables a equipos pequeños.

Las diferentes etapas de este proceso de maduración dentro de un contexto particular han sido analizadas por diferentes autores. Una de las primeras propuestas, elaborada por Martin y McClure [13], establece primordialmente tres fases: (1) Crisis y reconocimiento, (2) Énfasis Académico y (3) Asimilación y Madurez.

Siguiendo estos lineamientos, los centros de investigación deben realizar una estrategia de investigación que permita desarrollar y madurar las tecnologías a través del trabajo conjunto de universidades, grupos de investigación, empresas de la industria de software, y entidades gubernamentales relacionadas con el sector [2][23].

3.2 Consideraciones para los Tipos de Investigación

El Instituto de Ingeniería de Software [22] en Estados Unidos define tres etapas en su estrategia de investigación: (1) Creación, donde se identifican las tendencias y necesidades de la industria, y se maduran las tecnologías en el ámbito académico, (2) Aplicación, donde se aplica en entornos empresariales reales mediante proyectos de desarrollo y consultoría con el soporte del grupo de investigación para continuar madurando la tecnología, y (3) Amplificación, donde se definen estrategias de transferencia, como cursos, conferencias, libros y licencias de uso.

Algunos de los trabajos iniciales de Basili, Taschi y Zelkowitz [3][4] se centraron en el carácter experimental de los proyectos de investigación en Ingeniería de Software, y los diseños que deberían emplearse para su ejecución. Zelkowitz y Wallace definieron una clasificación de doce (12) modelos diferentes de verificación experimental en ingeniería de software [23][24].

Otros autores han propuesto clasificaciones sobre los tipos de investigaciones aplicados en Ingeniería de Software. Basili [3][4] propuso clasificar las investigaciones a partir de los experimentos que se definen en su interior: (1) "in vivo", cuando se desarrollan al interior de organizaciones que desarrollan software, y (2) "in vitro" cuando se realizan en entornos controlados. Kitchenham [10] por otra parte, clasificó los modelos experimentales en (1) cuantitativos, cuando se analizan variables cuantificables en los experimentos, (2) cualitativos, cuando se realizan revisión intersubjetiva de algunos atributos (por ejemplo, atributos de calidad) y (3) benchmarking, cuando se realizan pruebas comparativas de diferentes tecnologías para analizar su rendimiento y beneficio relativo.

Trabajos posteriores centrados en la estructura de los documentos técnicos que se presentan a las conferencias internacionales motivaron una serie de nuevas clasificaciones y consideraciones [9][21]. Shaw definió una serie de atributos de calidad que podrían evaluarse en un proyecto de investigación, así como una serie de recomendaciones para su diseño [18][20].

3.3 Tipos de Investigación

Para diseñar los diferentes proyectos de investigación, es necesario considerar el conjunto de elementos básicos de evaluación de la investigación. Una propuesta de elementos es definida por Shaw [19]: (1) el tipo de pregunta, (2) el producto final, y (3) el mecanismo de verificación.

De acuerdo a la clasificación de Shaw, el tipo de pregunta puede corresponder a: (1) Método de desarrollo, (2) Método de análisis o evaluación, (3) Diseño, evaluación o análisis de una instancia particular, (4) Generalización o caracterización, y (5) Estudio de factibilidad o exploración.

Según la misma clasificación el tipo de resultado de cada proyecto puede ser: (1) Procedimiento o técnica, (2) Modelo cualitativo o descriptivo, (3) Modelo empírico, (4) Modelo analítico, (5) Herramienta o notación, (6) Solución específica o prototipo, y (7) Reporte de experiencia.

El Mecanismo de validación, que garantiza la aplicabilidad de los resultados en otros ámbitos y empresas, puede ser: (1) Análisis, (2) Evaluación, (3) Experiencia, (4) Ejemplo, (5) Persuasión, y (6) Afirmación evidente. Los dos últimos mecanismos de verificación son inaceptables (o, por lo menos, los menos deseables) para los investigadores en el área.

El investigador, de acuerdo a la fase de la estrategia de investigación, debe seleccionar la combinación adecuada de los elementos de la investigación que le permita cumplir con los objetivos propuestos. Al conjugar el tipo de pregunta, el tipo de resultado y el mecanismo de validación, se configura el método o el diseño concreto de la investigación.

Con el fin diseñar sus proyectos, el investigador puede utilizar alguno de los arquetipos o patrones de diseño de investigación utilizados a nivel internacional. Zelkowitz y Wallace [23][24] definen tres categorías para los diferentes arquetipos de métodos de investigación y verificación en ingeniería de software: (1) Métodos de Observación, (2) Métodos Históricos, y (3) Métodos controlados.

Los métodos de Observación normalmente son mecanismos que permiten detectar aspectos interesantes al interior de proyectos de software y sirven de base para nuevos proyectos experimentales. Este tipo de métodos incluyen: (1) Monitoreo de proyecto, (2) Estudio de caso, (3) aserción, y (4) estudio de campo. Los métodos Históricos permiten recopilar experiencias y conocimientos de proyectos y productos ya elaborados, permitiendo reconocer y recopilar patrones y heurísticas, así como definir bases para nuevos proyectos experimentales. Los métodos Históricos son: (1) Investigación literaria, (2) datos legados, (3) lecciones aprendidas, y (4) análisis estático. Los métodos Controlados, están diseñados para establecer conclusiones fácilmente verificables, normalmente permitiendo corroborar descubrimientos, técnicas o modelos encontrados mediante las otras técnicas. Los métodos controlados son: (1) experimento replicado, (2) experimento en entorno sintético, (3) análisis dinámico, y (4) simulación, cuando se hacen simulaciones de los proyectos o productos en modelos del mundo real.

4 Propuesta de Estrategia de Investigación

Acorde con la concepción sobre el conocimiento, y sobre las formas de construcción de conocimiento en Ingeniería, se plantean algunas consideraciones y propuestas sobre las formas de investigación en Ingeniería de Software.

La evolución de la Ingeniería de Software no se ve limitada por los resultados presentados por grandes iniciativas de investigación. Laboratorios a nivel mundial invierten esfuerzos en el desarrollo de nuevo conocimiento tecnológico. Sin embargo, se deben realizar investigaciones en torno a proyectos reales de desarrollo de software. Las condiciones reales de una empresa de desarrollo de software son difícilmente reproducibles en laboratorio. De igual forma, la adaptación de métodos teóricos de ingeniería de software a procesos aplicables en organizaciones, es esencial para comprender las particularidades propias de los métodos y su aplicabilidad real en diferentes contextos. Por ejemplo, el mejoramiento de los procesos de desarrollo de software, a partir de procesos de diagnóstico y de rediseño de procesos, plantea posibilidades de creación de nuevos métodos y procedimientos para ser usados en diferentes empresas. Es así como, entre otros autores, Humphrey [8] logra realizar propuestas en torno a los procesos de desarrollo de software en equipos.

Como se ha revisado, en ocasiones los procesos de investigación en Ingeniería de Software surgen de la necesidad propia de la industria. En ese momento, o cuando se plantea de manera estructurada el desarrollo de una investigación, se debe analizar el contexto de aplicación que tendrá el conocimiento generado. Como en todo proyecto, se debe definir el alcance de la propuesta. Este alcance debe ser definido en términos de características preferiblemente cuantificables. Así, de manera natural se puede revisar la pertinencia y justificación del proyecto de investigación. Posteriormente, y luego de obtener resultados de la investigación, se debe validar y madurar el conocimiento construido, multiplicándolo en ambientes con contextos similares, siempre y hasta donde sea posible. La validación y maduración del conocimiento implica un proceso de inclusión de los conceptos en nuevos espacios. La inclusión de este conocimiento, mediante planes estructurados, permite no solo validar, sino ampliar el conocimiento hasta el momento construido.

De lo anterior, y basada en los modelos de Martin y McClure, y del SEI, se puede establecer una estrategia de investigación Esta consiste en tres grandes fases: (1) investigación y desarrollo inicial, (2) Investigación aplicada, y (3) Transferencia. Cada iniciativa o línea de investigación debe cumplir con estas tres fases, en el proceso de maduración de la tecnología. Cada una de ellas involucrando el desarrollo de varios proyectos de investigación, posiblemente cada uno de ellos con métodos y técnicas diferentes. La figura 1 presenta un esquema de la estrategia propuesta.

4.1 Investigación y desarrollo inicial

Representa el comienzo de cualquier iniciativa o línea de investigación. En ella se delimita el área de trabajo investigativo que se busca desarrollar. Normalmente surge a partir de una serie de lluvia de ideas en las cuales se busca determinar las principales necesidades de la industria o las tendencias de la tecnología que deben ser analizadas al interior del grupo. En algunos casos, puede surgir también a partir de una solución innovadora e interesante encontrada en la industria o por alguno de los investigadores. Para el desarrollo de esta etapa, normalmente se requieren varios tipos de investigaciones: diagnósticos, revisiones del estado del arte, investigaciones académicas y desarrollo de soluciones en el laboratorio (pruebas de concepto). Una iniciativa culminará esta etapa cuando pueda establecer, por lo menos en ambientes controlados, la factibilidad técnica de una solución al problema propuesto.

En algunos centros de investigación [22] existen procedimientos definidos para establecer las nuevas iniciativas o líneas de investigación. Algunas de estas, de hecho, pueden no superar la etapa inicial y no desarrollarse a través de proyectos piloto en empresas reales. En algunos casos se define una etapa de estudio de factibilidad antes de constituirse una nueva iniciativa.

4.2 Investigación aplicada

Es la etapa en donde se busca madurar la tecnología definida en la etapa inicial, trabajando en el desarrollo de la tecnología y en su aplicación en situaciones reales. A medida que se tienen experiencias de aplicación de la tecnología por parte del grupo de investigación y empresas de la industria, se encuentran las posibles fallas en el modelo de solución propuesto y se definen las adaptaciones que puedan ser requeridas para su aplicación en empresas. En esta etapa normalmente se desarrollan proyectos investigativos de naturaleza empírica y aplicada, en muchos casos siguiendo también esquemas de investigación-acción-participativa. Una iniciativa culminará esta etapa cuando pueda configurar una solución madura a alguna problemática en ingeniería de software, incluyendo consideraciones sobre las limitaciones e implicaciones de su implementación en empresas reales que realicen procesos de desarrollo de software.

4.3 Transferencia

Es la etapa que busca amplificar el impacto de las nuevas tecnologías, prácticas y conocimientos, más allá de las empresas con las cuales se trabajó en el proceso de investigación aplicada. Para cumplir su objetivo, los grupos de investigación desarrollan otros tipos de actividades, tales como cursos, conferencias, y licenciamiento de la tecnología.

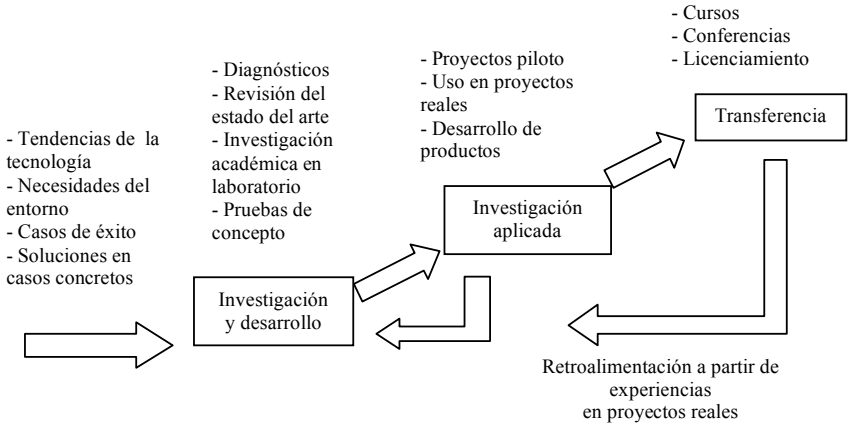


Figura 1. Esquema de la estrategia de investigación

5 RESULTADOS DE EXPERIENCIAS UTILIZANDO LA ESTRATEGIA DE INVESTIGACIÓN PROPUESTA

A partir de las consideraciones anteriores, el Laboratorio para el Desarrollo de la Ingeniería de Software (LIDIS) de la Universidad de San Buenaventura de Cali, Colombia, ha desarrollado una serie de actividades y proyectos de investigación al interior de empresas de desarrollo de software. El trabajo se realiza en dos contextos generales: al

interior de empresas medianas, y al interior de empresas pequeñas y nacientes, estas últimas típicamente instituciones incubadoras de empresas y parques tecnológicos.

La experiencia ha mostrado que las empresas grandes tienen características propias de definición de procesos, cultura organizacional y condiciones de burocracia que pueden dificultar el desarrollo de los procesos de innovación y el desarrollo de las investigaciones, sobre todo si la investigación ve afectado su nivel de producción del momento. Las empresas pequeñas, por otro lado, suelen tener muchos más problemas técnicos relacionados con el proceso de desarrollo, la construcción de productos genéricos y los procesos de mantenimiento y soporte. Por esto, se generan condiciones que son favorables para el desarrollo de investigaciones.

La primera iniciativa del grupo de investigación LIDIS en donde se ha aplicado parte de esta estrategia investigativa, se inició a finales de 2001 en el tema de "diseño y reutilización de arquitecturas de software". Esta iniciativa surgió con el fin de apoyar los procesos de fortalecimiento en temas de diseño de software y mejoramiento de productos de la red de Parques Tecnológicos de Software (ParqueSoft) y algunas empresas de desarrollo de software en Colombia.

Durante su desarrollo, la iniciativa ha incluido una gran variedad de proyectos, incluyendo la definición de dominios de aplicación, el diagnóstico sobre los productos y prácticas de desarrollo al interior de ParqueSoft, el establecimiento de nuevos estilos de arquitectura utilizando herramientas opensource, la documentación de patrones de diseño, la elaboración de librerías y marcos de trabajo (frameworks) y la construcción de herramientas de desarrollo y generación de código.

En su primera etapa, Investigación y Desarrollo Inicial, el grupo de investigadores ha trabajado en la configuración de proyectos que permitieran caracterizar mejor el problema, definir un contexto, y encontrar soluciones a los problemas iniciales. Muchas de estas investigaciones requirieron diseños variados: investigación literaria, estudio de campo, estudio de caso, experimento replicado y monitoreo de proyectos, entre otros. En esta etapa se han realizado actividades tanto en el laboratorio, como en las empresas mismas. Los proyectos usados para investigar en los problemas han permitido cubrir el más amplio espectro de posibilidades en términos de tipos de requerimientos funcionales y no funcionales aplicables al contexto. Al final de esta etapa se ha logrado definir un conjunto de procesos, métodos, técnicas, patrones, estilos y herramientas de software que podrían resolver el problema en una gran variedad de empresas en contextos similares.

En la segunda etapa de la estrategia, Investigación Aplicada, proyectos piloto fueron propuestos y realizados. En total se han desarrollado hasta el momento alrededor de siete proyectos piloto con empresas reales en donde los procesos, modelos, esquemas y herramientas propuestos se han colocado a prueba y han servido como retroalimentación a las investigaciones. Proyectos que han permitido constatar y comparar los resultados, y hacer modificaciones y mejoras sobre el conocimiento construido.

En la actualidad, se revisan los resultados de los proyectos piloto. Así, se refinan las soluciones planteadas, se definen modelos predecibles más confiables, y se establece una tecnología madura que pueda solucionar problemas, y aplicarse en otros contextos. Con esto se espera que los resultados del uso de la estrategia, luego de un número representativo de validaciones, generen reportes de experiencia y conocimiento en general que sea aceptado por la comunidad internacional en el área, como conocimiento aplicable.

6 CONCLUSIONES

La Ingeniería de Software es una profesión de origen tecnológico que aborda la construcción e implementación de un tipo particular de tecnología, y genera conocimientos relacionados con el desarrollo de estas prácticas en proyectos y situaciones reales. Plantear una estrategia de trabajo investigativo que considere el desarrollo de la tecnología, la puesta a prueba de la misma en situaciones reales, y su transferencia a las empresas de la industria de software, es un aspecto clave en el proceso de consolidación de los grupos y líneas de investigación.

El desarrollo de las investigaciones debe realizarse con un rigor apropiado, buscando mecanismos de verificación que posibiliten determinar la veracidad y aplicabilidad de sus resultados. Comprendiendo que los métodos no aplican indistintamente a todo tipo de proyectos u organizaciones, y que es necesario tener una mayor comprensión sobre la forma como ellos aplican en un caso determinado.

En la actualidad, una gran variedad de autores e investigadores han desarrollado algunos modelos y teorías relacionadas con los procesos de investigación en ingeniería de software. Lamentablemente, muchos de ellos parecen esquemas aislados o no articulados apropiadamente a la realidad de las necesidades del área.

Los grupos de investigación pueden establecer su propio esquema de trabajo a partir de la articulación de las diferentes teorías y modelos existentes, buscando establecer con claridad (1) la estrategia general para el trabajo investigativo, (2) las áreas temáticas que deben ser abordadas, (3) los elementos básicos del diseño de la investigación y (4) una serie de arquetipos o patrones de diseño de las investigaciones.

Basada [13] [22], el uso de una estrategia investigativa que incluya las tres fases de: (1) investigación y desarrollo inicial, (2) Investigación aplicada, y (3) Transferencia, puede ser el inicio de una estrategia adaptable a proyectos investigativos de Ingeniería de Software que permita la generación de conocimiento aceptado por la comunidad internacional en el área.

Finalmente, la definición de los contextos de aplicación del conocimiento generado no solo incluye características tecnológicas. Se debe considerar un proceso social que comprenda las actividades y relaciones que suceden al interior de las organizaciones, todas ellas gobernadas por principios de acción económica, política y axiológica.

Referencias

[1] ACOFI-ICFES. *Actualización y Modernización Curricular en Ingeniería de Sistemas*. ACOFI. Bogotá. Marzo 1996.

[2] *Consortium for Software Engineering Research*. *http://www.cser.ca*

[3] Basili, V. and R., *The Role of Experimentation: Past, Present, Future*, (Keynote presentation), 18th International Conference on Software Engineering, Berlin, Germany, March, 1996.

[4] Basili, V. *The experimental paradigm in software engineering* en Rombach, D., Basili, and V., Selby, R. *Experimental Software Engineering Issues: Critical Assessment and Future Directives*. Proceedings of Dagstuhl-Workshop. publicado en Lecture Notes in Computer Science #706. Springer-Verlag. 1993.

[5] Denning, P. *Educating a new Engineer*. Comunications ACM. Vol. 35. No. 12. Diciembre 1992.

[6] Finkelstein, A. and Kramer, J. *Software Engineering: a roadmap* en Finkelstein, A. (editor) Proceedings of the *Conference on The Future of Software Engineering*. ACM Press. 2002 *http://www.cs.ucl.ac.uk/staff/A.Finkelstein/fose/finalfinkelstein.pdf*

[7] Hilburn, T. and Hirmanpour, I. and Khaienoori, S. and Turner, R. and Qasem, A. *Software Engineering Body of Knowledge*. Versión 1.0. Software Engineering Institute. Carnegie Mellon. 1999.

[8] Humphrey, Watts. *Team Software Process*. Addison Wesley, 2000.

[9] Johnson, R. and Beck, K. and Booch. G. and Cook, W. and Gabriel, R. and Wirfs-Brock, R. *How to get a Paper Accepted at OOPSLA*. Panel. OOPSLA'93. *http://www.acm.org/sigs/sigplan/oopsla/oopsla96/how93.html*

[10] Kitchenham B. A., *Evaluating software engineering methods and tool*, ACM SIGSOFT Software Engineering Notes, (January, 1996) 11-15.

[11] Kirck, E. *Fundamientos de Ingeniería: Métodos, Conceptos y Resultados.* Limusa. México. 1979.

[12] Koen, B. *Definition of the Engineering Method*. American Society for Engineering Education. Washington. 1985.

[13] Martin, J. and McClure, C. *Structured Techniques for Computing*. Prentice-Hall / Englewood Cliffs, NJ, EE.UU. 1985

[14] Naur , P. and Randell, B. (editors) *Software Engineering: Report of a conference sponsored by the NATO Science Committee*, Garmish, Almania. Octubre 1968. NATO. *http://www.cs.ncl.ac.uk/old/people/brian.randell/home.formal/NATO/index.html*

[15] Parnas, D. *Software Engineering: An Unconsummed Marriage*. Communications of the ACM. Vol 40. No. 9. Septiembre 1997.

[16] Pfleeger, S. *Albert Einstein and Empirical Software Engineering*. Computer. Octubre 1999.

[17] Seaman, C. and Conradi, R. *Qualitative Methods in Software Engineering Research*. en la conferencia ISERN, Octubre 2000. *http://csdl.ics.hawaii.edu/isern/slides/seaman.qual.ppt*

[18] Shaw, M. *Designing Good Research Projects in Software Engineering… and getting results accepted for publication*. Carnegie Mellon University. *http://www.csc.calpoly.edu/~csturner/courses/shaw.pdf*

[19] Shaw, M. *What Makes Good Research in Software Engineering*. ETAPS. Grenoble, Suiza. 2002.

[20] Shaw, M. *Writing Good Software Engineering Research Papers*. Minitutorial. Internacional Conference on Software Engineering, ICSE 2003. Mayo 2003. *http://www-2.cs.cmu.edu/~Compose/shaw-icse03.pdf*

[21] Snyder, A. *How to get a Paper Accepted at OOPSLA*. OOPSLA'91 Proceedings. *http://www.acm.org/sigs/sigplan/oopsla/oopsla96/how91.html*

[22] *Software Engineering Institute*. Carnegie Mellon University. *http://www.sei.cmu.edu*

[23] Zelkowitz, M. and Wallace, D. *Experimental Models for Validating Technology*. IEEE Computer . Mayo 1998, pp 23-31.

[24] Zelkowitz, M. Wallace, D. *Experimental Validation in Software Engineering*. Conference of Empirical Assessment & Evaluation in Software Engineering, Keele University. Marzo 1997. *http://hissa.nist.gov/exper/ease.html*

Artículos XII Concurso Latinoamericano de Tesis de Maestria

InteGrade: Middleware para Computação em Grade Oportunista*

Andrei Goldchleger†, Fabio Kon
Departamento de Ciência da Computação
Universidade de São Paulo
{andgold,kon}@ime.usp.br
http://gsd.ime.usp.br/integrade

Abstract

Grid Computing allows for the integration of distributed computing resources, providing seamless access to their combined computing power. However, the existing Grid solutions targeted at low-end computing resources, such as desktop machines, have a number of limitations that hinder their usability in many application scenarios. This paper describes the core design of the InteGrade Grid Computing System, the implementation of various modules and tools, and the architecture and implementation of a parallel programming library that allows parallel applications to be executed on InteGrade grids.

Keywords: Grid Computing, Distributed Computing, Parallel Computing.

Resumo

A Computação em Grade permite a integração de recursos computacionais distribuídos, provendo acesso transparente à capacidade combinada de múltiplos recursos. Entretanto, as soluções para Computação em Grade voltadas a recursos computacionais de baixo custo apresentam uma série de limitações que impedem sua utilização por várias categorias de aplicação. Esse artigo descreve a arquitetura central do InteGrade, a implementação de diversos módulos e ferramentas, e a arquitetura e implementação de uma biblioteca de programação paralela que permite que aplicações paralelas sejam executadas em grades InteGrade.

Palavras-chave: Computação em Grade, Computação Distribuída, Computação Paralela.

1 Introdução

A computação se tornou uma ferramenta indispensável para as mais diversas atividades humanas. As ciências biológicas dependem de poder computacional para realizar tarefas como simulações, avaliação de modelos e mineração de dados. Físicos analisam grandes quantidades de dados geradas em experimentos realizados em aceleradores de partículas. A indústria cinematográfica utiliza aglomerados de computadores para gerar efeitos visuais cada vez mais realistas. A exploração de petróleo requer amplos estudos para determinar a probabilidade de existir petróleo em uma determinada localização. O mercado financeiro realiza simulações mercadológicas e análise de risco, o que requer uma grande capacidade de processamento. A cada dia, novas aplicações são concebidas, as quais requerem quantidades crescentes de computação.

A Computação em Grade [17] ajuda as instituições a lidar com a crescente necessidade de poder computacional: como é impossível obter capacidade de processamento adicional sem a aquisição de novos equipamentos, uma instituição pode implantar um sistema de Computação em Grade sobre os recursos que já possui, podendo utilizar sua capacidade combinada de processamento de maneira mais eficiente. Os sistemas de Grade facilitam as tarefas dos usuários e desenvolvedores de aplicações, pois provêem uma camada de abstração que encapsula a complexidade da infraestrutura distribuída, a qual potencialmente inclui recursos presentes em diferentes localizações geográficas e domínios administrativos.

Os sistemas de Computação em Grade podem ser divididos em duas grandes categorias. Alguns sistemas visam principalmente a integração de recursos computacionais de alto desempenho. Tais sistemas tendem a

*Este trabalho é financiado pelo CNPq, Brasil, processos 55.2028/2002-9 e 55.0094/2005-9.
†Andrei Goldchleger foi parcialmente financiado por uma bolsa de mestrado da CAPES, Brasil.

oferecer uma ampla gama de funcionalidades, como monitoramento de recursos, integração de computadores pertencentes a diversos domínios administrativos, e suporte a diversas categorias de aplicações paralelas. Entretanto, tais sistemas demandam uma implantação complexa, requerendo intervenções dos administradores de rede, além de equipamentos de alto custo. A segunda categoria objetiva integrar recursos computacionais de baixo custo, tais como computadores pessoais. Tais sistemas possuem implantação simplificada, porém oferecem funcionalidades limitadas. Em alguns casos, não possuem monitoramento de recursos, ou não permitem a execução de aplicações paralelas que demandam comunicação entre seus nós, o que impede sua utilização por determinadas categorias de aplicação.

O InteGrade [21] é um sistema de Computação em Grade que visa integrar recursos de baixo custo, tais como computadores pessoais. O InteGrade utiliza a capacidade ociosa de tais máquinas para executar aplicações da grade (daí o termo "Computação em Grade Oportunista"). O sistema oferece uma ampla gama de recursos que normalmente só estão presentes nos sistemas voltados a recursos de grande porte. Por exemplo, o sistema comporta a execução de diversas categorias de aplicação, incluindo aplicações seqüenciais (convencionais), *Bag-of-Tasks* (aplicações cujo domínio pode ser particionado sem que existam dependências entre cada partição), e paralelas com comunicação entre nós. A implementação e arquitetura do InteGrade são inteiramente orientadas a objetos, facilitando a integração de módulos e extensibilidade.

O trabalho realizado no contexto dessa dissertação de mestrado consistiu na definição da arquitetura básica do InteGrade, assim como na implementação dos seus principais módulos. Além disso, desenvolvemos uma biblioteca para programação paralela no InteGrade que permite a construção de aplicações que demandam comunicação entre seus nós. A organização deste artigo é a seguinte: a Seção 2 apresenta a arquitetura do InteGrade, sua implementação e avaliação de desempenho. A Seção 3 apresenta a biblioteca de programação paralela do InteGrade. Já a Seção 4 apresenta brevemente alguns trabalhos relacionados, e a Seção 5 apresenta as publicações produzidas durante o Mestrado. Finalmente, a Seção 6 apresenta nossas conclusões sobre o trabalho.

2 InteGrade

O Projeto InteGrade [33, 21] objetiva construir um middleware que permita a implantação de grades sobre recursos computacionais não dedicados, fazendo uso da capacidade ociosa normalmente disponível nos parques computacionais já instalados. É de conhecimento geral que grande parte dos computadores pessoais permanecem parcialmente ociosos durante longos períodos de tempo, culminando em períodos de ociosidade total: por exemplo, as estações de trabalho reservadas aos funcionários de uma empresa tradicional raramente são utilizadas à noite. Dessa maneira, a criação de uma infra-estrutura de software que permita a utilização efetiva de tais recursos que seriam desperdiçados possibilitaria uma economia financeira para as instituições que demandam grandes quantidades de computação. O InteGrade é um projeto desenvolvido conjuntamente por pesquisadores de três instituições: Departamento de Ciência da Computação (IME-USP), Departamento de Informática (PUC-Rio) e Departamento de Computação e Estatística (UFMS).

O InteGrade possui arquitetura orientada a objetos, onde cada módulo do sistema se comunica com os demais a partir de chamadas de método remotas. O InteGrade utiliza CORBA [27] como sua infra-estrutura de objetos distribuídos, beneficiando-se de um substrato elegante e consolidado, o que se traduz na facilidade de implementação, uma vez que a comunicação entre os módulos do sistema é abstraída pelas chamadas de método remotas. CORBA também permite o desenvolvimento para ambientes heterogêneos, facilitando a integração de módulos escritos nas mais diferentes linguagens, executando sobre diversas plataformas de hardware e software. Finalmente, CORBA fornece uma série de serviços úteis e consolidados, como os serviços de Transações [30], Persistência [29], Nomes [28] e *Trading*[1] [26], os quais podem ser utilizados pelo InteGrade, facilitando assim o desenvolvimento.

Desde a sua concepção, o InteGrade foi desenvolvido com o objetivo de permitir o desenvolvimento de aplicações para resolver uma ampla gama de problemas paralelos. Vários sistemas de Computação em Grade restringem seu uso a problemas que podem ser decompostos em tarefas independentes, como *Bag-of-Tasks* ou aplicações paramétricas. Alguns pesquisadores argumentam que sistemas de Computação em Grade não são apropriados para aplicações paralelas que possuem dependências entre seus nós, uma vez que tais dependências implicam em comunicações sobre redes de grande área, nem sempre robustas, e a aplicação

[1] O serviço de *Trading* definido em CORBA armazena *ofertas de serviços*, que contém características associadas a objetos CORBA. Através do uso da linguagem TCL (*Trader Constraint Language*), é possível realizar consultas de modo a obter referências a objetos que atendam a determinados requisitos.

inteira pode falhar por causa de um nó, caso não se adotem as medidas necessárias. De fato, algumas aplicações paralelas demandam as redes de interconexão proprietárias dos computadores paralelos, porém existe uma série de problemas que demandam comunicação e podem ser tratados por máquinas conectadas por redes convencionais. Além disso, nota-se a contínua evolução na capacidade de transmissão das redes locais e de grande área – o acesso do tipo "banda larga" já é uma realidade há anos, inclusive nas residências, e sua disseminação tende a aumentar, assim como a capacidade de transmissão de tais linhas.

O InteGrade é extremamente dependente dos usuários provedores de recursos, ou seja, usuários que compartilham a parte ociosa de seus recursos na Grade. Sem a colaboração de tais usuários, não existirão recursos disponíveis para as aplicações da Grade. Dessa maneira, o InteGrade pretende impedir que os usuários provedores de recursos sintam qualquer degradação de desempenho causada pelo InteGrade quando utilizam suas máquinas. Esse objetivo é atingido através da implantação de um módulo compacto nas máquinas provedoras de recursos, de maneira a consumir poucos recursos adicionais. Além disso, pretendemos utilizar o DSRT (*Dynamic Soft-Realtime CPU Scheduler*) [44], um escalonador em nível de aplicação para limitar a quantidade de recursos que pode ser utilizada pelas aplicações da Grade, permitindo assim que o usuário imponha políticas de compartilhamento de recursos, caso assim deseje. É importante notar que a aplicação de tais políticas é opcional, ou seja, mesmo que tais políticas não sejam aplicadas o InteGrade deve garantir a qualidade de serviço oferecida ao usuário provedor de recursos.

O InteGrade é um sistema muito dinâmico – uma vez que a maioria de seus recursos é compartilhada, a disponibilidade de recursos pode variar drasticamente ao longo do tempo e um recurso pode ser retomado por seu proprietário a qualquer momento. Tal ambiente é hostil às aplicações da Grade e prejudica o escalonamento, uma vez que as decisões de escalonamento podem ser invalidadas na prática devido à dinamicidade na disponibilidade dos recursos. Dessa maneira, o InteGrade pretende adotar um mecanismo para atenuar os efeitos do ambiente dinâmico: a Análise e Monitoramento dos Padrões de Uso, cujo objetivo é coletar longas séries de informações de maneira a permitir uma previsão probabilística da disponibilidade dos recursos compartilhados. Durante o escalonamento, a Análise e Monitoramento dos Padrões de Uso permitirá estimar por quanto tempo um recurso permanecerá ocioso, colaborando assim para melhores decisões de escalonamento.

Além das preocupações com desempenho, o InteGrade deve impedir que as máquinas da grade tenham sua segurança comprometida. Um usuário provedor de recursos deve estar seguro de que aplicações de terceiros submetidas através da Grade não comprometam seu sistema. Dessa maneira, o InteGrade deve tomar precauções para impedir que uma aplicação apague ou altere arquivos do usuário, ou tenha acesso a informações confidenciais. Uma abordagem a ser considerada é utilizar técnicas de *sandboxing* [20] para limitar as capacidades das aplicações de terceiros. Dessa maneira, por exemplo, podemos restringir o acesso ao sistema de arquivos a um determinado diretório, impedindo assim que as aplicações de terceiros obtenham acesso a dados confidenciais. Outro exemplo de uso é impedir que as aplicações da Grade acessem determinados dispositivos, como impressoras.

2.1 Arquitetura e Implementação

A arquitetura inicial do InteGrade foi inspirada no sistema operacional distribuído 2K [36]. Dessa maneira, o InteGrade herdou parte da nomenclatura de 2K. A unidade estrutural básica de uma grade InteGrade é o aglomerado (cluster). Um aglomerado é um conjunto de máquinas agrupadas por um determinado critério, como pertinência a um domínio administrativo. Tipicamente o aglomerado explora a localidade de rede, ou seja, o aglomerado contém máquinas que estão próximas uma das outras em termos de conectividade. Entretanto tal organização é totalmente arbitrária e os aglomerados podem conter máquinas presentes em redes diferentes, por exemplo. O aglomerado tipicamente contém entre uma e cem máquinas.

A Figura 1 apresenta os elementos típicos de um aglomerado InteGrade. Cada uma das máquinas pertencentes ao aglomerado também é chamada de nó, existindo vários tipos de nós conforme o papel desempenhado pela máquina. O Nó Dedicado é uma máquina reservada à computação em grade, assim como os nós de um aglomerado dedicado tradicional. Tais máquinas não são o foco principal do InteGrade, mas tais recursos podem ser integrados à grade se desejado. O Nó Compartilhado é aquele pertencente a um usuário que disponibiliza seus recursos ociosos à Grade. Já o Nó de Usuário é aquele que tem capacidade de submeter aplicações para serem executadas na Grade. Finalmente, o Gerenciador de Aglomerado é o nó onde são executados os módulos responsáveis pela coleta de informações e escalonamento, entre outros. Note que um nó pode pertencer a duas categorias simultaneamente – por exemplo, um nó que tanto compartilha seus recursos ociosos quanto é capaz de submeter aplicações para serem executadas na Grade.

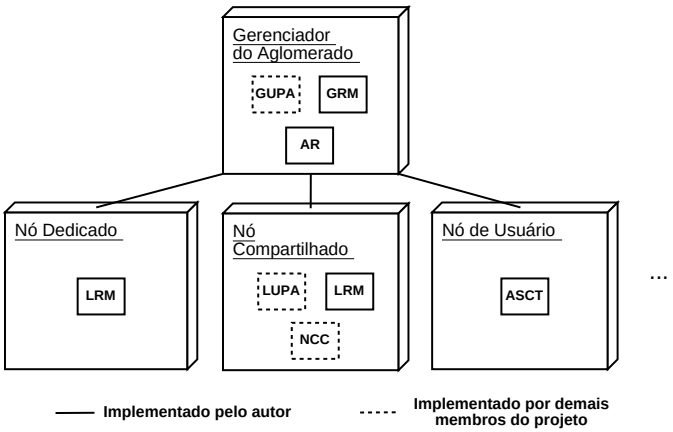


Figura 1: Arquitetura intra-aglomerado do InteGrade

Os módulos apresentados na Figura 1 são responsáveis pela execução de diversas tarefas necessárias à Grade. O **LRM** (Local Resource Manager) é executado em todas as máquinas que compartilham seus recursos com a Grade. É responsável pela coleta e distribuição das informações referentes à disponibilidade de recursos no nó em questão, além de exportar os recursos desse nó à Grade, permitindo a execução e controle de aplicações submetidas por usuários da Grade. Uma vez que o LRM é um módulo que deve consumir poucos recursos da máquina, escolhemos C++como a linguagem para seu desenvolvimento, aliada ao ORB compacto OiL [46], escrito na linguagem Lua [32].

O **GRM** (Global Resource Manager) tipicamente é executado no nó Gerenciador de Aglomerado e possui dupla função: (1) atua como o Serviço de Informações recebendo dos LRMs atualizações sobre a disponibilidade de recursos em cada nó do aglomerado e (2) atua como escalonador ao processar as requisições para a execução de aplicações na Grade. O GRM utiliza as informações de que dispõe para escalonar aplicações aos nós mais apropriados, conforme os requisitos das mesmas. A implementação do GRM utiliza a linguagem Java e o ORB JacORB [5].

O **NCC** (Node Control Center) é executado nas máquinas compartilhadas e permite que o proprietário da máquina controle o compartilhamento de seus recursos com a Grade. Atuando conjuntamente com o LRM, permite que o usuário defina períodos em que os recursos podem ou não ser utilizados (independente de estarem disponíveis), a fração dos recursos que pode ser utilizada (exemplo: 30% da CPU e 50% de memória) ou quando considerar a máquina como ociosa (por exemplo, quando não há de atividade de teclado por 5 minutos, ou quando o usuário encerra sua sessão). Convém lembrar que tais configurações são estritamente opcionais, uma vez que o sistema se encarregará de manter a qualidade de serviço provida ao proprietário dos recursos.

O **ASCT** (Application Submission and Control Tool) permite que um usuário submeta aplicações para serem executadas na Grade. O usuário pode estabelecer requisitos, como plataforma de hardware e software ou quantidade mínima de recursos necessários à aplicação, e preferências, como a quantidade de memória necessária para a aplicação executar com melhor desempenho. A ferramenta permite também que o usuário monitore e controle o andamento da execução da aplicação. Quando terminada a execução, o ASCT permite que o usuário recupere arquivos de saída de suas aplicações, caso existam, além do conteúdo das saídas padrão e de erro, úteis para diagnosticar eventuais problemas da aplicação. A implementação atual do ASCT consiste de uma aplicação gráfica escrita em Java utilizando JacORB.

O **LUPA** (Local Usage Pattern Analyzer) é responsável pela Análise e Monitoramento dos Padrões de Uso. A partir das informações periodicamente coletadas pelo LRM, o LUPA armazena longas séries de dados e aplica algoritmos de clustering [47] de modo a derivar categorias comportamentais do nó. Tais categorias serão utilizadas como subsídio às decisões de escalonamento, fornecendo uma perspectiva probabilística de maior duração sobre a disponibilidade de recursos em cada nó. Note que o LUPA só é executado em máquinas compartilhadas, uma vez que recursos dedicados são sempre reservados à computação em grade.

O **GUPA** (Global Usage Pattern Analyzer) auxilia o GRM nas decisões de escalonamento ao fornecer as informações coletadas pelos diversos LUPAs. O GUPA pode funcionar de duas maneiras, de acordo com

as políticas de privacidade do aglomerado – se as informações fornecidas pelo LUPA não comprometem a privacidade de um usuário, como é o caso de uma estação de trabalho de um laboratório, o GUPA pode agir como aglomerador das informações disponibilizadas pelos LUPAs. Entretanto, caso o perfil de uso gerado pelo LUPA comprometa a privacidade do proprietário de um recurso (como uma estação de trabalho pessoal), os padrões de uso nunca deixam o LUPA – nesse caso, o GUPA realiza consultas específicas ao LUPA, podendo assim funcionar como *cache* das respostas fornecidas sob demanda pelos diversos LUPA.

O **AR** (Application Repository) armazena as aplicações a serem executadas na Grade. Através do ASCT, o usuário registra a sua aplicação no repositório para posteriormente requisitar sua execução. Quando o LRM recebe um pedido para execução de uma aplicação, o LRM requisita tal aplicação ao Repositório de Aplicações. O Repositório de Aplicações pode fornecer outras funções mais avançadas, como por exemplo o registro de múltiplos binários para a mesma aplicação, o que permite executar uma mesma aplicação em múltiplas plataformas, a categorização de aplicações, facilitando assim a busca por aplicações no repositório, a assinatura digital de aplicações, o que permite verificar a identidade de quem submeteu a aplicação, e controle de versões. Assim como o GRM, o Repositório de Aplicações foi desenvolvido em Java e utiliza o JacORB para a comunicação com os demais módulos.

2.1.1 Protocolo de Disseminação de Informações

O Protocolo de Disseminação de Informações do InteGrade permite que o GRM mantenha informações relativas à disponibilidade de recursos nas diversas máquinas do aglomerado. Tais informações incluem dados estáticos (arquitetura da máquina, versão do sistema operacional e quantidades totais de memória e disco) e dinâmicos (porcentagem de CPU ociosa, quantidades disponíveis de disco e memória, entre outros). Tais informações são importantes para a tarefa de escalonamento de aplicações; de posse de uma lista de requisitos da aplicação, o GRM pode determinar uma máquina adequada para executá-la. O Protocolo de Disseminação de Informações do InteGrade é o mesmo utilizado pelo sistema operacional distribuído 2K [36].

Uma questão importante associada ao Protocolo de Disseminação de Informações é a periodicidade com a qual as atualizações são feitas. Se atualizarmos as informações muito freqüentemente, o desempenho do sistema tende a degradar, uma vez que a rede será tomada por mensagens de atualização. Por outro lado, quanto maior o intervalo de atualização de informações, maior a tendência de tais informações não corresponderem à real disponibilidade de recursos nas máquinas do aglomerado. Dessa maneira utilizamos o conceito de dica (*hint*) [37], ou seja, as informações mantidas no GRM fornecem uma visão aproximada da disponibilidade de recursos no aglomerado.

O Protocolo de Disseminação de Informações é o seguinte: periodicamente, a cada intervalo de tempo t_1, o LRM verifica a disponibilidade de recursos do nó. Caso tenha havido uma mudança significativa entre a verificação anterior e a atual, o LRM envia tais informações ao GRM. A determinação do que é significativo é dada através de uma porcentagem para a qual uma mudança é considerada significativa (por exemplo, 10% de variação na utilização de CPU). Entretanto, caso não hajam mudanças significativas em um intervalo t_2 ($t_2 > t_1$), o LRM mesmo assim manda uma atualização das informações. Tal atualização serve como *keep-alive* e permite que o GRM detecte quedas dos LRM. Essa abordagem resulta em redução do tráfego na rede, uma vez que as mensagens só são enviadas no caso de mudanças significativas, ou em intervalos maiores, no caso de servirem como *keep-alive*.

2.1.2 Protocolo de Execução de Aplicações

O Protocolo de Execução de Aplicações do InteGrade permite que um usuário da grade submeta aplicações para execução sobre recursos compartilhados. Assim como o Protocolo de Disseminação de Informações, o Protocolo de Execução é derivado do protocolo utilizado no sistema 2K, porém alterado para o InteGrade. A Figura 2 ilustra os passos do protocolo. Uma vez que a aplicação tenha sido registrada no Repositório de Aplicações através do ASCT, o usuário solicita a execução da aplicação (1). O usuário pode, opcionalmente, especificar requisitos para a execução de sua aplicação, como por exemplo a arquitetura para a qual a aplicação foi compilada, ou a quantidade mínima de memória necessária para a execução. Também é possível especificar preferências, como executar em máquinas mais rápidas, por exemplo.

Assim que a requisição de execução é enviada ao GRM, este procura um nó candidato para executar a aplicação (2). Utilizando os requisitos da aplicação informados pelo usuário e as informações sobre disponibilidade de recursos nos nós fornecidas pelo Protocolo de Disseminação de Informações já descrito, o GRM procura por um nó que possua recursos disponíveis para executar a aplicação. Caso nenhum nó satisfaça

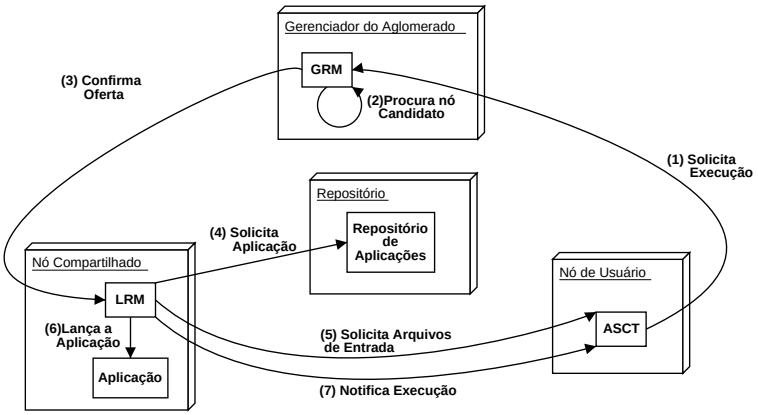


Figura 2: Protocolo de Execução de Aplicações

os requisitos da aplicação, o GRM notifica tal fato ao ASCT que solicitou a execução. Entretanto, caso haja algum nó que satisfaça os requisitos, o GRM envia a solicitação para o LRM da máquina candidata a executar a aplicação (3). Nesse momento, o LRM verifica se, de fato, possui recursos disponíveis para executar a aplicação – como já mencionado, o GRM mantém uma visão aproximada da disponibilidade de recursos nos nós do aglomerado, o que leva à necessidade de tal verificação. Caso o nó não possua os recursos necessários para a execução da aplicação, o LRM notifica o GRM de tal fato e o GRM retorna ao passo (2), procurando por outro nó candidato.

Entretanto, caso o nó candidato possa executar a aplicação, o LRM de tal nó solicita a aplicação em questão ao Repositório de Aplicações (4), solicita os eventuais arquivos de entrada da aplicação ao ASCT requisitante (5), e lança a aplicação (6), notificando ao ASCT que sua requisição foi atendida (7). O ASCT assim descobre em qual LRM sua aplicação está executando, podendo assim controlá-la remotamente.

Quando a requisição de execução envolve uma aplicação composta por múltiplos nós, por exemplo, uma aplicação paralela com n nós, o protocolo é levemente alterado: no passo (2), o GRM procura até n máquinas candidatas a executar a aplicação[2]. Posteriormente, no passo (3), o GRM realiza n chamadas para executar a aplicação, cada uma destas solicitando a execução de um dos nós da aplicação.

Atualmente, a implementação do protocolo encontra-se incompleta: quando o GRM envia a requisição para o LRM, este ainda não verifica se tem recursos disponíveis para executar a aplicação, portanto, atualmente, um LRM nunca recusa um pedido de execução. Essa é uma deficiência que deve ser sanada no futuro.

2.2 Avaliação de Desempenho do LRM

O LRM é certamente o módulo mais crítico do InteGrade no tocante ao consumo de recursos. Como é executado em máquinas compartilhadas, o LRM não pode utilizar muitos recursos computacionais, sob pena de causar degradação na qualidade de serviço oferecida aos usuários que compartilham suas máquinas com a Grade. Assim, realizamos alguns experimentos com o objetivo de medir a utilização de CPU e memória do LRM. Os experimentos foram realizados em uma máquina com processador Athlon XP 2800+ e 1 GiB de memória RAM. A máquina utilizada possui o sistema operacional GNU/Linux (Debian 3.1 *testing*), kernel 2.6.10 e biblioteca de *threads* com suporte a NPTL (*Native POSIX Thread Library*) [14]. É importante notar que a combinação do kernel 2.6.10 com a biblioteca NPTL é fundamental [48] para permitir a correta medição da utilização de recursos de aplicações *multi-threaded* como o LRM. Utilizamos a ferramenta `top` para a medição do consumo de CPU e `pmap` para obter o consumo detalhado de memória de uma aplicação. Ambas ferramentas fazem parte do pacote Procps. Para a determinação do consumo de memória efetivo da aplicação (RSS – *Resident Set Size*)[3], inspecionamos o pseudo-diretório `/proc` através de *scripts*.

[2]Caso não consiga n máquinas, o GRM pode escalonar dois ou mais nós da aplicação para uma mesma máquina.

[3]O RSS representa a quantidade de memória física que uma determinada aplicação ocupa em um determinado momento. É a soma do espaço ocupado pelo executável, pilha, bibliotecas compartilhadas e área de dados.

A Tabela 1 apresenta o uso de memória do LRM após sua iniciação[4]. Note que a maioria do espaço ocupado em memória pelo LRM se refere a bibliotecas compartilhadas. Do total de bibliotecas, apenas a `liblua` (80 KiB) e a `liblualib` (88 KiB) são bibliotecas de uso específico. As demais bibliotecas são de uso geral e muito provavelmente são utilizadas por aplicações do usuário, já em execução na mesma máquina e, portanto, não implicam em gastos adicionais. Assim, dos 2816 KiBs ocupados após a iniciação, apenas 576 KiB são exclusivos do LRM, o que é pouco quando consideramos que as estações de trabalho atuais possuem ao menos 256 MiB de memória RAM. Tais observações nos levam a concluir que a implementação do LRM atingiu o objetivo de gastar poucos recursos adicionais das máquinas compartilhadas.

Consumo de Memória do LRM		
Tipo	Tamanho (KiB)	
Resident Set Size	2816	
Executável	128	
Pilha	52	
Bibliotecas	2408	
Consumo de Memória por Biblioteca		
Nome	Finalidade	Tamanho (KiB)
libc	Biblioteca padrão C	1188
libstdc++	Biblioteca padrão C++	636
libm	Biblioteca matemática C	132
ld	Carregador de bibliotecas dinâmicas	88
liblualib	Biblioteca Lua	88
liblua	Biblioteca Lua	80
libresolv	Resolução de nomes (DNS)	60
libpthread	Biblioteca de *threads*	48
libnss_files	Obtenção de informações de configuração do sistema	36
libgcc_s	Biblioteca de suporte a exceções para C++	32
libnss_dns	Obtenção de informações de configuração do sistema	12
libdl	Biblioteca para carga dinâmica de bibliotecas	8

Tabela 1: Consumo detalhado de memória do LRM

O primeiro experimento realizado refletiu a operação do LRM quando este não recebe nenhuma requisição para executar aplicações. Nesse cenário, o LRM apenas envia periodicamente para o GRM atualizações sobre a quantidade de recursos disponíveis no nó, além de verificar periodicamente a terminação de aplicações previamente iniciadas. Configuramos o LRM de maneira a verificar a variação da disponibilidade de recursos a cada segundo e enviar uma atualização ao GRM a cada dois segundos no máximo. Para realizar as medições, utilizamos um programa especialmente desenvolvido para coletar a utilização de CPU e memória (*Resident Set Size*) a cada três segundos. O programa `top` foi utilizado para fins de controle, permitindo a validação dos dados obtidos por nosso programa. O monitoramento foi realizado durante 5 minutos.

A Figura 3 apresenta os resultados obtidos no experimento. Podemos notar que o consumo de CPU se mantém extremamente baixo, em menos de 1%, sendo que na maioria do tempo a utilização de CPU permaneceu em 0%. Através do programa desenvolvido, verificamos que a quantidade total de CPU utilizada (`usertime + systemtime`) aumenta muito lentamente, sendo tais aumentos extremamente pequenos. Quanto ao uso de memória, podemos notar que sofre um pequeno aumento que não chega a ser significativo.

O segundo experimento reflete o consumo de recursos quando o LRM recebe requisições de execução. Nesse experimento, o LRM recebia uma requisição para a execução e lançava uma instância de uma aplicação seqüencial. Esse procedimento foi repetido cinqüenta vezes, com um intervalo de dez segundos entre cada requisição. As medições foram realizadas com as mesmas ferramentas utilizadas no primeiro experimento. A Figura 4 apresenta os gráficos dos resultados obtidos pelo experimento. Notamos que o consumo de CPU para lançar uma instância da aplicação é extremamente baixo, sempre permanecendo inferior a 2%. Já o consumo de memória cresce rapidamente após as primeiras requisições e fica sujeito a oscilações nas

[4]Apesar de incluído no RSS, o espaço destinado à área de dados não consta na tabela pois pode variar de tamanho ao longo da execução do LRM.

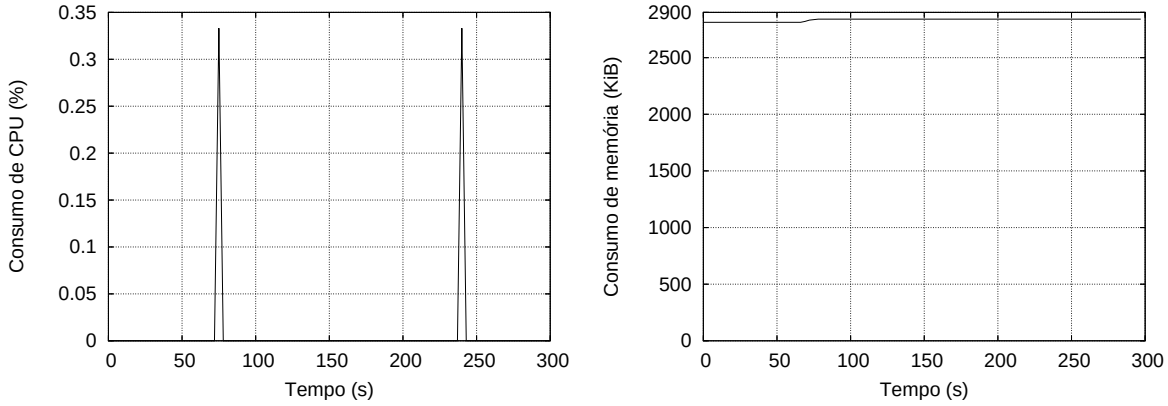


Figura 3: Experimento 1: Consumo de CPU e memória do LRM

execuções seguintes. Entretanto, o crescimento no consumo de memória do LRM não chega a consumir uma quantidade significativa de recursos da máquina. No futuro, experimentos adicionais podem ser conduzidos de maneira a determinar mais precisamente os motivos que levam ao aumento no consumo de memória.

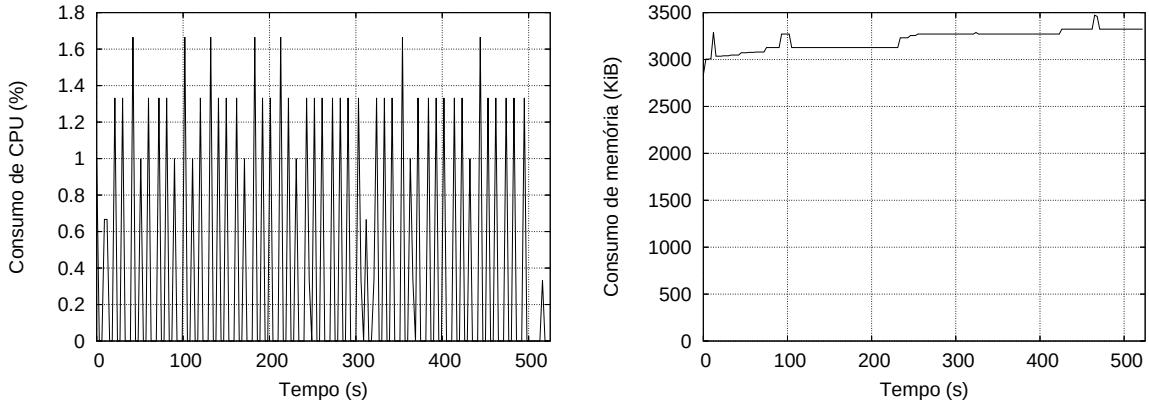


Figura 4: Experimento 2: Consumo de CPU e memória do LRM

3 A Biblioteca de Programação Paralela

Ambientes de Computação em Grade possuem diversas características que os tornam extremamente adequados à execução de aplicações paralelas. A grande disponibilidade de recursos sugere a possibilidade de executarmos várias aplicações paralelas sobre os mesmos, eliminando a necessidade de recursos dedicados. Porém, após uma análise inicial, nos deparamos com diversos problemas relacionados à execução de aplicações paralelas sobre as Grades:

- *comunicação*: algumas aplicações paralelas fortemente acopladas demandam grande quantidade de comunicação entre seus nós. Tais aplicações dificilmente podem se beneficiar de sistemas de Computação em Grade, a não ser em casos excepcionais onde seja possível a utilização de redes de altíssima velocidade. Ainda assim, existe uma ampla gama de aplicações que podem ser executadas em ambientes típicos de Grade;
- *tolerância a falhas*: executar uma aplicação paralela sobre uma grade cria dificuldades inexistentes quando trabalhamos com máquinas paralelas ou aglomerados dedicados. O ambiente de Grade, devido

a sua natureza altamente distribuída, é muito mais propenso a falhas. Dessa maneira, sistemas de Computação em Grade idealmente precisam prover mecanismos de tolerância a falhas adequados, de maneira a garantir que aplicações progridam em sua execução, mesmo em caso de falhas;

- *checkpointing*: a habilidade de salvar o estado da aplicação de maneira a garantir o seu progresso é uma característica importante que deve estar presente nos sistemas de Computação em Grade, especialmente em ambientes que fazem uso oportunista de recursos ociosos, como o InteGrade. Porém, *checkpointing* de aplicações paralelas é um problema significativamente mais difícil que *checkpointing* de aplicações convencionais [55];

- *variedade de modelos existentes*: existe uma ampla gama de modelos e implementações de bibliotecas para programação paralela, dentre as quais podemos citar MPI [25], PVM [51], BSP [54] e CGM [13], entre outros. Tais bibliotecas foram criadas anteriormente à existência de sistemas de Computação em Grade, portanto existe uma considerável quantidade de aplicações já escritas para as mesmas. Assim, é importante que os sistemas de Computação em Grade ofereçam suporte a tais bibliotecas, de maneira a permitir que aplicações pré-existentes possam ser executadas sobre as grades sem que sejam necessários grandes esforços para modificar tais aplicações.

Apesar dos problemas, o suporte a aplicações paralelas está presente nos principais sistemas de Computação em Grade. A abordagem mais usada é intuitiva: cria-se uma biblioteca que possua as mesmas interfaces de uma já existente fora do contexto de grade, como MPI. Internamente, tal biblioteca possui uma implementação específica para cada sistema de Computação em Grade, porém tais detalhes são ocultados do usuário, que enxerga apenas as mesmas interfaces com as quais estava acostumado a programar na biblioteca original. Tal abordagem é bem sucedida, uma vez que permite utilizar aplicações existentes sobre as Grades efetuando poucas ou até mesmo nenhuma alteração.

Legion provê suporte a aplicações MPI e PVM [45]. Aplicações pré-existentes precisam apenas ser recompiladas e religadas de maneira a permitir que se beneficiem das facilidades oferecidas pelo sistema. Uma alternativa possível é executar aplicações MPI e PVM sem recompilá-las, porém facilidades como *checkpointing* não estarão disponíveis para as aplicações.

Condor também provê suporte a MPI e PVM. O suporte a PVM de Condor não requer que a aplicação seja recompilada e religada, mas a aplicação PVM deve seguir o paradigma mestre-escravo. Dessa maneira, aplicações PVM existentes podem ser executadas diretamente sobre Condor, respeitando as restrições de arquitetura e sistema operacional. Já o suporte a MPI apresenta um inconveniente: as máquinas sobre as quais as aplicações executam devem ser designadas "reservadas", ou seja, quando começam a executar uma aplicação paralela tais máquinas o fazem até o fim, não podendo sofrer assim nenhum tipo de interrupção [55]. Tal inconveniente dificulta o uso de máquinas compartilhadas e computação oportunista na execução de aplicações paralelas.

Globus provê suporte para aplicações escritas em MPI, através da biblioteca MPICH-G2 [35], uma implementação de MPI específica para o Globus, porém compatível com as demais implementações de MPI. MPICH-G2 utiliza o Globus para a comunicação entre os nós das aplicações, provendo diversos protocolos de comunicação e conversão no formato de dados. Mais recentemente, Globus foi estendido para prover suporte ao modelo BSP [53, 52]. A arquitetura proposta é similar à disponível para execução de programas MPI.

3.1 O Modelo BSP

O modelo BSP (*Bulk Synchronous Parallelism*) [54] foi concebido como uma nova maneira de se escrever programas paralelos. Assim como ocorreu nos casos do MPI e PVM, o modelo BSP é genérico o suficiente de maneira que possa ser implementado sobre as mais diferentes arquiteturas, desde uma máquina paralela até um aglomerado de PCs conectados por meio de uma rede *Ethernet*. Entretanto, o modelo BSP e suas implementações se diferenciam de modelos como o MPI ou o PVM. Primeiramente, as bibliotecas que implementam o modelo BSP costumam ser enxutas: por exemplo, a BSPlib [31], uma das implementações do BSP, possui em sua biblioteca núcleo apenas 20 funções[5]. A característica fundamental do modelo BSP é o desacoplamento entre comunicação e sincronização entre os nós de uma aplicação paralela. Outra característica do modelo BSP é a possibilidade de se fazer uma análise prévia do custo de execução dos programas: o modelo BSP provê métricas para o cálculo do desempenho de programas em uma certa arquitetura. Baseando-se

[5] A biblioteca MPI é composta por mais de 100 funções.

em parâmetros como o número de processadores, o tempo necessário para sincronizar os processadores e a razão entre as vazões de comunicação e computação é possível calcular previamente o desempenho de um programa.

Teoricamente, o modelo BSP é composto por um conjunto de processadores virtuais, cada qual com sua memória local. Tais processadores virtuais são conectados por uma rede de comunicação. No caso de uma máquina paralela, os processadores virtuais são mapeados para os processadores da máquina e a rede de comunicação é a infra-estrutura específica de interligação dos processadores em tal arquitetura, podendo ser um barramento, por exemplo. No caso de um aglomerado de computadores, cada "processador virtual" é um computador e a rede de comunicação é a rede local *Ethernet*, por exemplo.

O mecanismo de comunicação entre os nós de uma aplicação BSP não é restringido pelo modelo. Exemplos de mecanismos presentes nas implementações do modelo são: memória distribuída compartilhada e troca de mensagens (*Message Passing*).

A estrutura de um programa BSP é composta por uma série de *superpassos* (*supersteps*). Cada superpasso é composto por 3 etapas: inicialmente, cada uma das tarefas do programa realiza computação com os dados que possui localmente. Posteriormente, todas as comunicações pendentes entre as tarefas são realizadas. Finalmente, ocorre uma barreira de sincronização que marca o fim de um superpasso e o início do seguinte. Uma característica importante é o fato da comunicação só ser efetivada *no final* do superpasso. Ou seja, se uma tarefa escreve um valor na memória de outra tarefa, tal escrita só se materializa de fato no próximo superpasso. Algumas implementações, como a BSPlib, oferecem métodos que permitem a escrita e leitura imediata de valores em outras tarefas, porém tais métodos devem ser usados com cuidado e seguindo uma série de restrições de maneira a garantir a correção da aplicação.

3.2 A Implementação

Um dos objetivos da implementação BSP no InteGrade é permitir que aplicações BSP existentes possam ser executadas sobre a Grade com um mínimo de alterações. Apesar de compartilharem o mesmo modelo, cada biblioteca que o implementa possui interfaces diferentes, prejudicando assim a portabilidade. Para minimizar tal inconveniente, decidimos utilizar em nossa implementação a interface C[6] da BSPlib, uma das implementações BSP disponíveis. Assim, a tarefa de portar uma aplicação que utiliza a BSPlib para o InteGrade consiste apenas em incluir um cabeçalho diferente, recompilar a aplicação e religá-la com a nossa biblioteca BSP. Tal facilidade no porte é muito importante, uma vez que aplicações existentes podem se beneficiar dos recursos de uma grade sem que seja necessário um processo de porte trabalhoso e caro.

Outra característica importante da implementação BSP no InteGrade é a sua relativa independência em relação ao resto do sistema. Como InteGrade é um sistema em desenvolvimento intenso, é importante que suas interfaces mantenham-se enxutas, descrevendo apenas a funcionalidade essencial do sistema. Dessa maneira, toda a funcionalidade relativa à implementação do BSP é descrita em interfaces IDL separadas das demais. As interfaces IDL dos módulos do InteGrade em sua maioria não foram alteradas: a única alteração, realizada no ASCT, consistiu na adição de um método.

A biblioteca BSP do InteGrade utiliza CORBA para a comunicação entre os nós da aplicação, facilitando o desenvolvimento, manutenção e extensão do código. O uso de CORBA pode ser questionado em uma aplicação de alto desempenho como uma biblioteca para programação paralela, porém no caso específico de nossa implementação existem alguns atenuantes: (1) InteGrade se beneficia de poder de processamento que estaria ocioso, portanto, desempenho neste caso não é a preocupação principal, uma vez que ele provavelmente vai ser comprometido por outros fatores, como a necessidade de migração resultante da indisponibilidade de recursos. (2) ORBs compactos são utilizados com êxito em ambientes altamente restritivos no tocante à disponibilidade de recursos, como sistemas embutidos. Além disso, experimentos com o UIC-CORBA [49] mostram uma perda de desempenho de apenas 15% quando comparado a soquetes, demonstrando que é possível combinar as vantagens de CORBA a um desempenho bastante razoável. Caso necessário, no futuro, poderemos utilizar apenas soquetes ao custo de aumentar a complexidade da biblioteca BSP do InteGrade.

As aplicações BSP no InteGrade normalmente são compostas por dois ou mais nós, também chamados de processos ou tarefas. Cada nó possui um identificador único dentro da aplicação, o *BSP PID*, utilizado por exemplo no endereçamento das comunicações para envio de dados – todas as funções da biblioteca para tal finalidade possuem como um de seus parâmetros o identificador da tarefa para a qual a chamada deve ser feita. É importante ressaltar que as aplicações BSP são do tipo SPMD (*Single Program, Multiple Data*), ou

[6] Existe também uma interface para FORTRAN na BSPlib, porém não a oferecemos no InteGrade.

seja, todos os nós da aplicação executam o mesmo programa. É comum, entretanto, que diferentes nós da aplicação executem diferentes trechos de código – por exemplo, é comum que apenas um nó realize tarefas de totalização de resultados. Os identificadores de processo são novamente úteis nesse cenário: pode-se por exemplo determinar que um dado trecho de código seja executado apenas pelo nó que possua um determinado identificador.

As aplicações BSP em muitos aspectos são tratadas pelo InteGrade de maneira similar às aplicações convencionais: por exemplo, o registro de uma aplicação BSP no Repositório de Aplicações se dá de maneira idêntica ao registro de uma aplicação convencional. As requisições de execução referentes a aplicações BSP são tratadas pelo GRM de maneira idêntica às aplicações paramétricas: para todos os efeitos, uma aplicação BSP é apenas uma aplicação que possui múltiplos nós. É bem provável que futuramente as diferentes classes de aplicações venham a ser tratadas de maneira diferente: por exemplo, como os nós das aplicações BSP comunicam-se entre si, é desejável que eles sejam alocados para máquinas com boa conectividade. Entretanto, ao minimizar a necessidade de mudanças para cada classe de aplicação contribuímos para a simplicidade do sistema, introduzindo novas características apenas quando necessário.

Cada uma das tarefas da aplicação BSP possui um *BspProxy* associado, um servente CORBA que recebe mensagens relacionadas à execução da aplicação BSP. O BspProxy é criado de maneira independente em cada nó, durante a iniciação da aplicação. O BspProxy representa o lado servidor de cada um dos nós da aplicação BSP, recebendo mensagens de outros processos, tais como escritas ou leituras remotas em memória, mensagens sinalizando o fim da barreira de sincronização, entre outras. A Figura 5 apresenta a interface IDL do BspProxy. A finalidade de cada método será descrita posteriormente, no contexto das funções da biblioteca BSP do InteGrade.

```
interface BspProxy{
  void registerRemoteIor(in long pid, in string ior);
  void takeYourPid(in long pid);
  void bspPut(in types::DrmaOperation drmaOp);
  void bspGetRequest(in types::BspGetRequest request);
  void bspGetReply(in types::BspGetReply reply);
  void bspSynch(in long pid);
  void bspSynchDone(in long pid);
};
```

Figura 5: Interface IDL do BspProxy

Em determinados momentos, as aplicações BSP necessitam de um coordenador central. As principais tarefas que demandam coordenação são:

- *difusão do endereço IOR das tarefas*: como cada uma das tarefas de uma aplicação BSP potencialmente comunica-se com as demais, é conveniente que cada uma das tarefas possa se comunicar diretamente com as outras. Dessa maneira, é necessário que o coordenador colete e posteriormente distribua os endereços IOR de todas as tarefas que compõem a aplicação;
- *atribuição de identificadores de processo*: cada nó da aplicação deve ser identificado unicamente. Assim, o coordenador deve ser responsável por atribuir identificadores a cada uma das tarefas que compõem a aplicação;
- *coordenação das barreiras de sincronização*: as barreiras de sincronização ao final de cada superpasso indicam que cada uma das tarefas atingiu um determinado ponto em sua execução. O coordenador é responsável por receber as mensagens das tarefas que atingiram a barreira, e quando todas a tiverem atingido, o coordenador deve notificar que cada uma das tarefas pode prosseguir em sua execução.

Em nossa implementação, decidimos que o coordenador seria um dos nós da própria aplicação, dispensando assim serviços externos de coordenação. Dessa maneira, elegemos um dos nós da aplicação para ser o coordenador da mesma. Esse nó é denominado *Process Zero* em alusão ao seu identificador de processo – tal nó é responsável por atribuir os identificadores de processo e ele sempre atribui zero a si próprio. Note que a escolha do coordenador é feita de maneira totalmente transparente ao programador da aplicação. Além

disso, o Process Zero não tem suas atividades restritas à coordenação, ele executa normalmente suas tarefas como os demais nós da aplicação.

A implementação dos métodos da BSPlib no InteGrade se encontra parcialmente completa. Até o presente momento, implementamos as funções necessárias para a iniciação de uma aplicação BSP, algumas funções de consulta sobre características da aplicação, as funções que permitem a comunicação entre processos através de memória compartilhada distribuída e a função de sincronização dos processos. As funções que permitem comunicação por troca de mensagens (*Bulk Synchronous Message Passing* (BSMP)) foram implementadas por Carlos Alexandre Queiroz, membro do projeto InteGrade; entretanto, tais funções não serão aqui apresentadas. Nas seções seguintes, apresentaremos os métodos implementados e o seu funcionamento interno, assim como as principais classes que compõem a biblioteca.

3.2.1 Funções de Iniciação e Consulta

A Figura 6 apresenta parte das funções básicas presentes na BSPlib e que já se encontram implementadas na biblioteca BSP do InteGrade. A função `bsp_begin` determina o início do trecho paralelo de uma aplicação BSP. De acordo com as especificações da BSPlib, cada programa BSP deve ter apenas um trecho paralelo[7]. Nenhuma das demais funções da biblioteca pode ser chamada antes de `bsp_begin`, uma vez que esta realiza as tarefas de iniciação da aplicação, entre elas a eleição do Process Zero, o coordenador da aplicação.

```
void bsp_begin(int maxProcs)
int bsp_pid()
int bsp_nprocs()
void bsp_end()
```

Figura 6: Funções básicas da biblioteca BSP do InteGrade

A eleição de um nó coordenador para a aplicação implica em um conhecimento mútuo dos nós participantes da eleição. Entretanto, convém lembrar que como os nós de uma aplicação BSP são escalonados de maneira semelhante às demais aplicações, cada nó da aplicação não conhece os demais. Dessa maneira, tornou-se necessário utilizar um intermediário que seja conhecido por todos os nós da aplicação, de maneira que estes possam descobrir uns aos outros. Ao invés de implementarmos um serviço especial para tal tarefa, optamos por estender a interface do ASCT para realizar a tarefa de iniciação da aplicação. Uma vez que o ASCT é o requisitante das execuções, foi possível convertê-lo facilmente em um serviço conhecido por todos os nós da aplicação, de maneira a permitir a descoberta mútua.

Ao realizar uma requisição que envolva uma aplicação BSP, o ASCT adiciona um arquivo especial na lista de arquivos de entrada da aplicação, o `bspExecution.conf`. A Figura 7 apresenta um exemplo do conteúdo desse arquivo. O campo `appMainRequestId` contém um identificador da aplicação BSP emitido pelo ASCT. O campo `asctIor` contém o endereço IOR do ASCT que requisitou a execução e o campo `numExecs` indica quantos nós fazem parte da aplicação. Esse arquivo é transferido ao LRM que atendeu à requisição da mesma maneira que os demais arquivos de entrada.

```
appMainRequestId 3
asctIor IOR:00CAFEBA...
numExecs 8
```

Figura 7: Exemplo de arquivo bspExecution.conf

A primeira tarefa realizada por `bsp_begin` é a eleição do Process Zero. O processo de eleição é extremamente simples: a partir do endereço IOR do ASCT contido em `bspExecution.conf`, `bsp_begin` instancia um *stub* para o ASCT e realiza a chamada `registerBspNode`, contendo como parâmetros o identificador da aplicação no ASCT e o endereço IOR do nó BSP em questão[8]. Cada nó da aplicação realiza tal chamada de maneira independente. O nó que tiver sua chamada completada primeiro é automaticamente eleito

[7] O trecho paralelo da aplicação, delimitado pelas chamadas `bsp_begin` e `bsp_end`, pode ser precedido ou seguido por trechos seqüenciais de código, ou seja, código `C` que não utiliza as funções da biblioteca BSP.

[8] Esse endereço IOR é o endereço do BspProxy associado ao nó, criado no início de `bsp_begin`.

coordenador. Nessa situação, o valor de retorno de `registerBspNode` é o *struct* `BspInfo` com os campos preenchidos da seguinte forma: `isProcessZero` é verdadeiro, `processZeroIor` é o endereço IOR do próprio processo. Para os demais nós, `BspInfo` é devolvido com `isProcessZero` igual a falso e `processZeroIor` contendo o endereço IOR do nó que foi eleito coordenador.

Após a eleição do Process Zero, restam ainda dois passos de iniciação da aplicação: a distribuição de identificadores de processo BSP e a difusão dos endereços IOR de cada tarefa que compõe a aplicação. A Figura 8 apresenta um diagrama de seqüência de tais passos envolvendo o Process Zero e uma das demais tarefas[9]. Cada nó da aplicação realiza os seguintes passos: envia seu endereço IOR para o Process Zero, através do método `registerRemoteIor` em BspProxy, e bloqueia até receber o seu identificador de processo BSP e os endereços IOR das demais tarefas. Já o Process Zero, a cada endereço IOR recebido, atribui e envia um identificador de processo BSP para a tarefa em questão, através do método `takeYourPid`. Quando todas as tarefas realizaram tais passos, o Process Zero envia para *cada tarefa* os pares *(BSP PID, IOR)* de *todas* as tarefas que fazem parte da aplicação. Dessa maneira, nas operações subseqüentes, cada tarefa pode comunicar-se diretamente com as demais, dispensando intermediários.

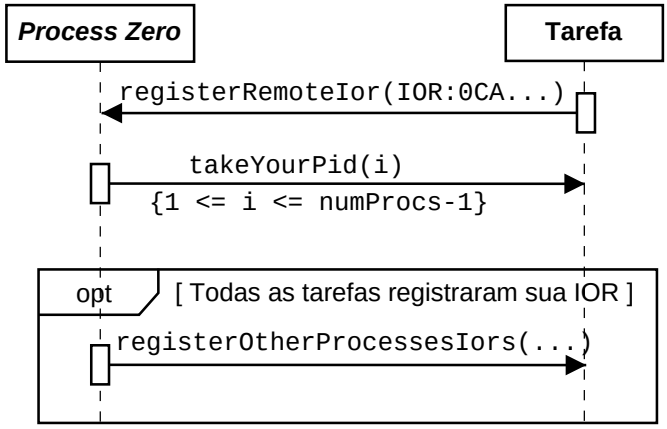


Figura 8: Diagrama de seqüência das tarefas de iniciação da aplicação BSP

A função `bsp_pid`, apresentada na Figura 6, fornece o identificador de processo BSP da tarefa em questão. Tal função é comumente usada para indicar que apenas uma determinada tarefa realize uma função na aplicação: por exemplo, pode-se assim determinar que apenas a tarefa de identificador i gere arquivos de saída. Já a função `bsp_nprocs` devolve o número de tarefas que compõem a aplicação BSP. Finalmente, a função `bsp_end` delimita o final do bloco paralelo da aplicação. Em nossa implementação, esse método não precisa realizar nenhuma operação.

3.2.2 Distributed Remote Memory Access (DRMA)

Distributed Remote Memory Access (DRMA) é um dos métodos disponíveis na BSPlib para a comunicação entre os processos de uma aplicação BSP. DRMA provê a abstração de memória compartilhada distribuída: determinadas áreas de memória de cada nó de uma aplicação podem ser acessadas pelos demais nós para operações de leitura e escrita.

Antes de realizar uma operação de leitura ou escrita na memória compartilhada, todos os nós da aplicação devem registrar a área de memória envolvida com a biblioteca BSP. Cada nó da aplicação BSP possui uma Pilha de Registros, como a exibida na Figura 9, onde são guardados registros descrevendo as áreas de memória que podem ser envolvidas em operações de escrita e leitura. Cada registro contém um endereço físico de memória e um tamanho em bytes referente ao tamanho da área de memória registrada. A posição do registro na Pilha de Registros representa o *endereço lógico* do registro. Uma determinada variável X definida na aplicação BSP certamente terá diferentes endereços físicos em cada um dos nós da aplicação, porém possuirá o mesmo endereço lógico em todos os nós da aplicação.

[9]Ou seja, as atividades descritas no diagrama se repetem para cada um dos demais nós da tarefa.

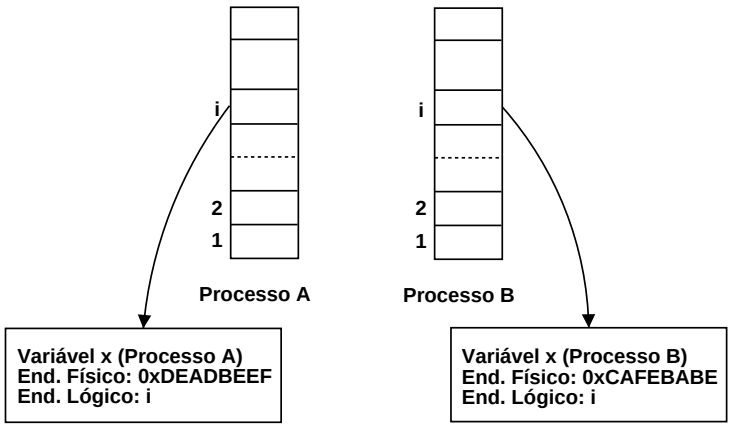


Figura 9: Exemplo da Pilha de Registros

A partir da Figura 9, podemos derivar um exemplo de como funciona a resolução de endereços com o auxílio da Pilha de Registros. Suponha que o processo A deseja escrever ou ler a variável X no processo B. Como se trata de uma aplicação distribuída, A não tem como diretamente obter o endereço físico de X em B. Assim, A consulta sua Pilha de Registros à procura de algum registro que contenha a variável X, buscando pelo endereço físico local de X. Tal busca é feita a partir do topo da pilha, decrescendo eventualmente até a base. Quando A acha algum registro para a variável X, obtém dessa maneira o endereço lógico de X, i, que é único para todos os processos. Dessa maneira, A envia mensagem a B indicando a escrita ou leitura na variável de endereço lógico i. B consulta a posição i sua Pilha de Registros, descobrindo assim o endereço físico local de X, podendo assim realizar a operação de escrita ou leitura. Note a importância de que o endereço lógico de uma variável na pilha seja o mesmo em todos os processos: caso contrário, uma escrita na variável X poderia resultar em uma escrita na variável Z, causando erros no programa.

A Figura 10 apresenta o conjunto de funções para DRMA da BSPlib que já se encontram implementadas na biblioteca BSP do InteGrade. O método `bsp_pushregister` registra na pilha uma posição de memória de endereço `addr` e tamanho `size` em bytes. Note que é possível registrar um endereço de memória múltiplas vezes – a resolução de endereços é sempre realizada a partir do topo da pilha, dessa maneira, para um dado endereço físico de memória o registro devolvido será o mais recentemente registrado. Tal característica é útil quando são manipuladas estruturas compostas, como vetores, que podem ser registrados com diferentes comprimentos caso seja conveniente. Já o método `bsp_popregister` remove da pilha um registro referente ao endereço `addr`. Note que o método não recebe o tamanho do registro como parâmetro: a pilha é percorrida a partir do topo em direção a base e o primeiro registro encontrado referente a `addr` é removido. É importante ressaltar que a inserção e remoção de registros só será efetivada no final do superpasso.

```
void bsp_pushregister(const void * addr, int size)
void bsp_popregister(const void * addr)
void bsp_put(int pid, const void * src, void * dst, int offset, int nbytes)
void bsp_get(int pid, const void * src, int offset, void * dst, int nbytes)
```

Figura 10: Funções da biblioteca BSP do InteGrade para comunicação DRMA

O método `bsp_put` permite que um nó da aplicação realize uma escrita remota na memória de outra tarefa que compõe a aplicação. O parâmetro `pid` contém o identificador do processo no qual será realizado a escrita. `src` aponta para os dados a serem copiados para o processo remoto. `dst` é o endereço físico local da variável onde a escrita vai ser efetivada. Tal endereço será traduzido para o endereço lógico da variável pelo procedimento já descrito. Note que `dst` deve ser um endereço registrado na Pilha de Registros. Já o parâmetro `offset` representa um deslocamento a partir do endereço `dst`, ou seja, os dados serão escritos a partir do endereço `dst + offset`. Finalmente, `nbytes` indica o número de bytes que serão copiados. A

chamada remota (`bspPut` do BspProxy) é realizada no momento da chamada a `bsp_put`, porém a escrita no processo remoto só é efetivada ao final do superpasso.

De maneira oposta a `bsp_put`, o método `bsp_get` permite que um determinado nó da aplicação BSP leia um valor da memória de outra tarefa da aplicação. O método `bsp_get` possui uma assinatura semelhante à de `bsp_put`: o parâmetro `pid` contém o identificador do processo BSP do qual se lerão os dados. `src` aponta para os dados a serem copiados a partir do processo remoto. Tal endereço será traduzido para o endereço lógico da variável pelo procedimento já descrito. `offset` representa um deslocamento a ser aplicado a partir do endereço `src`, ou seja, os dados serão copiados a partir do endereço `src + offset`. É importante ressaltar que `src` deve estar registrado na Pilha de Registros. Já `dst` representa o endereço físico local para onde os dados serão copiados. Finalmente, `nbytes` indica o número de bytes que serão copiados a partir de `src + offset`. Assim como ocorre com `bsp_put`, a chamada remota (`bspGetRequest` do BspProxy) é realizada no momento da chamada a `bsp_get`, porém a leitura só é efetivada ao final do superpasso: após a barreira de sincronização, o processo alvo lê o valor requisitado e o envia ao requisitante através do método `bspGetReply`.

3.2.3 A Barreira de Sincronização

Conforme citado previamente, a computação no modelo BSP se desenvolve em termos da unidade básica chamada superpasso. Durante um superpasso, cada processo pode registrar e excluir posições da Pilha de Registros, assim como escrever ou ler resultados na memória dos demais, porém todas as operações só são efetivadas após todos os processos atingirem a barreira de sincronização. O método chamado em cada processo para indicar que uma barreira de sincronização foi atingida é `void bsp_sync()`.

Em nossa implementação, a barreira de sincronização é coordenada pelo Process Zero. Quando um processo executa a função `bsp_sync` presente no código de uma aplicação BSP, tal chamada desencadeia uma chamada remota `bspSynch` ao BspProxy associado ao Process Zero. Nesse momento, o processo que atingiu a barreira tem sua execução bloqueada, ficando apenas recebendo eventuais mensagens dos outros processos. Já o Process Zero, ao receber `bspSynch`, contabiliza o número de processos que já enviaram tal mensagem. Caso todos os processos tenham enviado tal mensagem, e inclusive o Process Zero tenha atingido a barreira, o Process Zero envia a mensagem `bspSynchDone` para todos os processos. Ao receberem tal mensagem, todos os processos são desbloqueados e passam a tratar as mensagens recebidas e os eventos ocorridos durante o superpasso, sempre na seguinte ordem:

1. `bsp_get`: todas as requisições de leitura efetuadas por outros processos são atendidas ao final do superpasso. Note que os valores são retornados antes de sofrer quaisquer eventuais alterações originadas pelos demais processos, uma vez que os `bsp_get` são tratados `antes` dos `bsp_put`;
2. `bsp_put`: as escritas remotas realizadas pelos demais processos são então efetivadas;
3. Registros de variáveis na pilha, ou seja, chamadas a `bsp_pushregister`;
4. Exclusões de variáveis da pilha, ou seja, chamadas a `bsp_popregister`.

4 Trabalhos Relacionados

Nos últimos anos, a pesquisa em Computação em Grade foi muito ativa, resultando em uma grande quantidade de sistemas para as mais diversas finalidades. A Tabela 2 apresenta uma comparação entre os diversos sistemas aqui apresentados.

O Globus [19, 16] é o projeto de maior impacto na área de Computação em Grade. O principal foco do projeto é a integração de recursos computacionais de alto desempenho. Baseado no conceito de caixa de ferramentas (*Globus Toolkit* – GT), permite que as Grades e suas respectivas aplicações sejam construídas de maneira incremental através da progressiva adição de serviços. A versão 3.0 (GT3) marcou uma grande mudança em relação a anterior (GT2), caracterizada pela adoção de Web Services como tecnologia de integração dos serviços da grade, chamados de *Grid Services*, além da definição e implementação de diversos padrões tais como OGSA [15], OGSI [18] e WSRF [8], que oferecem diretrizes para a construção de serviços da Grade.

O Legion [38, 24] objetivou a construção de um sistema de Computação em Grade totalmente orientado a objetos. Construiu uma infraestrutura baseada em Objetos Núcleo [40], modelo de objetos distribuídos que

proviam funcionalidades básicas aos demais serviços da grade. Apesar de possuir os mesmos objetivos do Globus, a arquitetura de Legion era radicalmente diferente: enquanto cada serviço Globus era construído de maneira independente, os serviços Legion eram todos construídos sobre seu arcabouço de objetos distribuídos. De certa maneira, a versão 3.0 de Globus adotou parte dos aspectos arquiteturais de Legion.

Condor [7, 41] é o pioneiro dos sistemas de Computação em Grade. Assim como o InteGrade, objetiva a utilização do poder de processamento ocioso de equipamentos de baixo custo. Inicialmente concebido como um sistema de gerenciamento de um aglomerado de máquinas, evoluiu de maneira a integrar recursos pertencentes a diversos domínios administrativos, permitindo inclusive a integração com grades Globus. Condor provê *checkpointing* para aplicações convencionais, permitindo que uma execução interrompida a qualquer momento seja retomada posteriormente. É possível utilizar Condor para executar aplicações paralelas MPI e PVM, porém estas últimas devem seguir o modelo de mestre-escravos. Entretanto, Condor não provê *checkpointing* de aplicações paralelas.

MyGrid [43, 6] permite que grades computacionais sejam implantadas facilmente pelo próprio usuário sobre recursos aos quais tem acesso. A implantação do sistema é simplificada, requerendo a instalação de alguns poucos módulos. MyGrid permite a execução de aplicações do tipo *Bag-of-Tasks*, mas não comporta aplicações que exigem comunicação entre nós. Devido a sua simplicidade, também não implementa alguns recursos típicos de uma grade, como monitoramento de recursos. OurGrid [3] é uma extensão de MyGrid que permite o compartilhamento de recursos computacionais em uma rede *Peer-to-Peer* [39]. Dessa maneira, a integração de recursos passa a ser maior, uma vez que é possível integrar recursos pertencentes a diversos indivíduos e instituições.

SETI@home [50, 2] consiste de uma aplicação distribuída com o objetivo de analisar ondas de rádio em busca de indícios que sugiram inteligência extra-terrestre. Distribuído no formato de um protetor de tela, a aplicação periodicamente se conecta a um complexo de servidores centralizados, os quais enviam um conjunto de dados a serem processados. Dessa maneira, o projeto utiliza a capacidade de processamento ociosa de milhares de computadores pessoais distribuídos ao redor do mundo. Apesar de suas limitações arquiteturais, o projeto alcançou grande êxito no seu objetivo de arregimentar colaboradores, possuindo mais de quatro milhões de usuários cadastrados e cerca de seiscentos mil usuários ativos.

BOINC (*Berkeley Open Infrastructure for Network Computing*) [4, 1] é um projeto sucessor de SETI@home, e objetiva sanar parte das limitações de seu antecessor. BOINC permite que a mesma aplicação cliente (o protetor de tela) realize processamento de dados para diversos projetos, cuja participação é definida pelo usuário que cede seus recursos. Apesar das melhorias, BOINC ainda limita seu uso à execução de aplicações do tipo *Bag-of-Tasks*.

Característica \ Sistema	GT2	GT3	Legion	Condor	MyGrid	OurGrid	SETI@home	BOINC	InteGrade
Grade computacional tradicional	✗	✗	✗		✗	✗			
Grade computacional oportunista				✗			✗	✗	✗
Código aberto	✗	✗			✗	✗		✗	✗
Binários gratuitos	✗	✗		✗	✗	✗	✗	✗	✗
Implementação orientada a objetos		✗	✗		✗	✗		✗	✗
Comunicação baseada em padrões[a]	✗[b]	✗							✗
Suporte a múltiplas aplicações	✗	✗	✗	✗	✗	✗		✗	✗
Suporte a aplicações paralelas[c]	✗	✗	✗	✗					✗
Implementa o padrão OGSA		✗							

[a] Comunicação baseada em padrões da indústria tais como CORBA e Web Services.

[b] Apenas em alguns serviços, entre eles o MDS.

[c] Consideramos apenas aplicações paralelas que exigem comunicação entre seus nós, ou seja, que não sejam trivialmente paralelizáveis.

Tabela 2: Comparação entre diversos sistemas de Computação em Grade

5 Publicações

O trabalho aqui representado gerou oito publicações; em três delas participei como autor principal:

- *InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines* [22]: artigo curto publicado no *ACM/IFIP/USENIX Middleware'2003 International Workshop on Middleware for Grid Computing*, descreve a arquitetura e os primeiros esforços de implementação do InteGrade;
- *InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines* [21]: artigo publicado no periódico *Concurrency and Computation: Practice & Experience*, é uma versão estendida do artigo anterior;
- *Running Highly-Coupled Parallel Applications in a Computational Grid* [23]: artigo curto publicado no 22^o *Simpósio Brasileiro de Redes de Computadores (SBRC)*, descreve a implementação da primeira versão da biblioteca BSP do InteGrade.

Participei como co-autor nas seguintes publicações:

- *Grid: An Architectural Pattern* [10]: artigo publicado na *11th Conference on Pattern Languages of Programs (PLoP'2004)*, é um padrão arquitetural que descreve os sistemas de Computação em Grade de maneira genérica, listando os principais serviços que estes oferecem;
- *Grid Middleware: Leveraging Distributed Processing Capabilities* [9]: versão estendida do padrão anterior, a ser publicado no livro *Pattern Languages of Program Design (PloPD)*;
- *Checkpointing-based Rollback Recovery for Parallel Applications on the InteGrade Grid Middleware* [12]: artigo publicado no *ACM/IFIP/USENIX Middleware'2004 2nd International Workshop on Middleware for Grid Computing*, descreve a versão inicial do mecanismo de *checkpointing* para aplicações paralelas desenvolvidas com a biblioteca BSP do InteGrade;
- *Checkpointing-based Rollback Recovery for Parallel Applications on the InteGrade Grid Middleware* [11]: artigo a ser publicado no periódico *Concurrency and Computation: Practice & Experience*, é uma versão estendida do artigo anterior;
- *InteGrade: A Tool for Executing Parallel Applications on a Grid for Opportunistic Computing* [34]: Artigo publicado no Salão de Ferramentas do 23^o *Simpósio Brasileiro de Redes de Computadores (SBRC)*, descreve em detalhes as ferramentas do InteGrade que permitem a execução e monitoramento de aplicações.

6 Conclusão

O trabalho aqui apresentado e desenvolvido durante este mestrado consistiu na definição da arquitetura básica e implementação da fundação do sistema InteGrade. O resultado de tal trabalho produziu um sistema totalmente funcional que pode ser utilizado para a execução de diferentes classes de aplicações. A biblioteca de programação paralela permite a execução de aplicações paralelas que demandam comunicação entre nós, aumentando assim a gama de problemas que podem se beneficiar da grade.

A infra-estrutura aqui descrita serve como fundação para a pesquisa de outros membros do projeto, que resultará em novas funcionalidades para o sistema. As áreas de pesquisa do InteGrade incluem segurança, tolerância a falhas, escalonamento, agentes móveis e algoritmos paralelos. Os resultados estão sendo progressivamente incorporados à base de código do InteGrade.

Além de servir como substrato para a pesquisa dos demais membros do projeto, o trabalho aqui desenvolvido é utilizado como ponto de partida para dois projetos independentes: o projeto MAG [42] utiliza agentes móveis para a resolução de problemas que demandam grandes quantidades de computação; já o projeto FlexiGrid visa investigar diversos aspectos referentes à construção de grades computacionais, tais como qualidade de serviço, mobilidade e adaptação dinâmica.

O InteGrade é um projeto de código aberto hospedado na Incubadora Virtual da FAPESP, no endereço `http://incubadora.fapesp.br/projects/integrade`. A incubadora permite que qualquer interessado obtenha o código fonte do sistema e sua documentação. O texto completo da dissertação de mestrado está disponível no endereço `http://gsd.ime.usp.br/publications/thesis/MestradoAndrei.pdf`.

Referências

[1] David P. Anderson. BOINC: A System for Public-Resource Computing and Storage. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 4–10. IEEE Computer Society, 2004.

[2] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky e Dan Werthimer. SETI@home: an Experiment in Public-Resource Computing. *Communications of the ACM*, 45(11):56–61, 2002.

[3] Nazareno Andrade, Walfredo Cirne, Francisco Brasileiro e Paulo Roisenberg. OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing. In *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, volume 2862, pages 61–86. Springer Verlag, Junho de 2003. Lecture Notes in Computer Science.

[4] BOINC. `http://boinc.berkeley.edu`. Último acesso em Julho/2005.

[5] Gerald Brose. JacORB: Implementation and Design of a Java ORB. In *Proceedings of the DAIS'97, IFIP WG 6.1 International Working Conference on Distributed Aplications and Interoperable Systems*, pages 143–154, Setembro de 1997.

[6] Walfredo Cirne, Daniel Paranhos, Lauro Costa, Elizeu Santos-Neto, Francisco Brasileiro, Jacques Sauvé, Fabrício A. B. Silva, Carla O. Barros e Cirano Silveira. Running Bag-of-Tasks Applications on Computational Grids: The MyGrid Approach. In *Proceedings of the 2003 International Conference on Parallel Processing*, pages 407–416, Outubro de 2003.

[7] Condor. `http://www.cs.wisc.edu/condor`. Último acesso em Julho/2005.

[8] Karl Czajkowski, Don Ferguson, Ian Foster, Jeff Frey, Steve Graham, Tom Maguire, David Snelling e Steve Tuecke. *From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution*, Março de 2004. Version 1.1.

[9] Raphael Y. de Camargo, Andrei Goldchleger, Marcio Carneiro e Fabio Kon. Grid Middleware: Leveraging Distributed Processing Capabilities. A ser publicado no livro *Pattern Languages of Program Design 5 (PloPD'5)*.

[10] Raphael Y. de Camargo, Andrei Goldchleger, Marcio Carneiro e Fabio Kon. Grid: An Architectural Pattern. In *The 11th Conference on Pattern Languages of Programs (PLoP'2004)*, Monticello, Illinois, USA, Setembro de 2004.

[11] Raphael Y. de Camargo, Andrei Goldchleger, Fabio Kon e Alfredo Goldman. Checkpointing-based Rollback Recovery for Parallel Applications on the InteGrade Grid Middleware. A ser publicado no periódico *Concurrency and Computation: Practice and Experience*.

[12] Raphael Y. de Camargo, Andrei Goldchleger, Fabio Kon e Alfredo Goldman. Checkpointing-based Rollback Recovery for Parallel Applications on the InteGrade Grid Middleware. In *Proceedings of the ACM/IFIP/USENIX Middleware'2004 2nd International Workshop on Middleware for Grid Computing*, pages 35–40, Toronto, Ontario, Canada, Outubro de 2004.

[13] F. Dehne. Coarse grained parallel algorithms. *Algorithmica Special Issue on "Coarse grained parallel algorithms"*, 24(3–4):173–176, 1999.

[14] Ulrich Drepper e Ingo Molnar. The Native POSIX Thread Library for Linux. White Paper, Red Hat, Fevereiro de 2003.

[15] I. Foster, C. Kesselman, J. Nick e S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Junho de 2002. Global Grid Forum, Open Grid Service Infrastructure Working Group.

[16] Ian Foster e Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 2(11):115–128, 1997.

[17] Ian Foster e Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., 2003.

[18] Global Grid Forum. *Open Grid Services Infrastructure (OGSI) Version 1.0*, Junho de 2003. GFD-R-P.15 (Proposed Recommendation).

[19] Globus. `http://www.globus.org`. Último acesso em Julho/2005.

[20] Ian Goldberg, David Wagner, Randi Thomas e Eric A. Brewer. A Secure Environment for Untrusted Helper Applications: Confining the Wily Hacker. In *Proceedings of the 6th Usenix Security Symposium*, pages 1–13, San Jose, CA, USA, Julho de 1996.

[21] Andrei Goldchleger, Fabio Kon, Alfredo Goldman, Marcelo Finger e Germano Capistrano Bezerra. InteGrade: object-oriented Grid middleware leveraging the idle computing power of desktop machines. *Concurrency and Computation: Practice and Experience*, 16(5):449–459, Março de 2004.

[22] Andrei Goldchleger, Fabio Kon, Alfredo Goldman vel Lejbman e Marcelo Finger. InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines. In *Proceedings of the ACM/IFIP/USENIX Middleware'2003 1st International Workshop on Middleware for Grid Computing*, pages 232–234, Rio de Janeiro, Junho de 2003.

[23] Andrei Goldchleger, Carlos Alexandre Queiroz, Fabio Kon e Alfredo Goldman. Running Highly-Coupled Parallel Applications in a Computational Grid. In *Proceedings of the 22th Brazilian Symposium on Computer Networks (SBRC' 2004)*, Gramado-RS, Brazil, Maio de 2004. Short paper.

[24] Andrew S. Grimshaw, Wm. A. Wulf e The Legion Team. The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, 40(1):39–45, 1997.

[25] W. Gropp, E. Lusk, N. Doss e A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, Setembro de 1996.

[26] Object Management Group. *Trading Object Service Specification*, Junho de 2000. OMG document formal/00-06-27, version 1.0.

[27] Object Management Group. *CORBA v3.0 Specification*. Needham, MA, Julho de 2002. OMG Document 02-06-33.

[28] Object Management Group. *Naming Service Specification*, Setembro de 2002. OMG document formal/02-09-02, version 1.2.

[29] Object Management Group. *Persistent State Service Specification*, Setembro de 2002. OMG document formal/02-09-06, version 2.0.

[30] Object Management Group. *Transaction Service Specification*, Setembro de 2003. OMG document formal/03-09-02, version 1.4.

[31] Jonathan M. D. Hill, Bill McColl, Dan C. Stefanescu, Mark W. Goudreau, Kevin Lang, Satish B. Rao, Torsten Suel, Thanasis Tsantilas e Rob H. Bisseling. BSPlib: The BSP programming library. *Parallel Computing*, 24(14):1947–1980, 1998.

[32] Roberto Ierusalimschy, Luiz Henrique de Figueiredo e Waldemar Celes Filho. Lua-an extensible extension language. *Software: Practice & Experience*, 26:635–652, 1996.

[33] InteGrade. `http://gsd.ime.usp.br/integrade`. Último acesso em Julho/2005.

[34] José Braga Pinheiro Jr., Raphael Y. de Camargo, Andrei Goldchleger e Fabio Kon. InteGrade: a Tool for Executing Parallel Applications on a Grid for Opportunistic Computing. In *Proceedings of the 23th Brazilian Symposium on Computer Networks (SBRC Tools Track)*, Fortaleza-CE, Brasil, Maio de 2005.

[35] N. Karonis, B. Toonen e I. Foster. MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing (JPDC)*, 63(5):551–563, Maio de 2003.

[36] Fabio Kon, Roy H. Campbell, M. Dennis Mickunas, Klara Nahrstedt e Francisco J. Ballesteros. 2K: A Distributed Operating System for Dynamic Heterogeneous Environments. In *Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing (HPDC '9)*, pages 201–208, Pittsburgh, Agosto de 2000.

[37] Butler W. Lampson. Hints for computer system design. In *Proceedings of the Ninth ACM Symposium on Operating Systems Principles*, pages 33–48. ACM Press, 1983.

[38] Legion. `http://www.cs.virginia.edu/~legion`. Último acesso em Julho/2005.

[39] Bo Leuf. *Peer to Peer: Collaboration and Sharing over the Internet*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[40] Michael J. Lewis e Andrew Grimshaw. The Core Legion Object Model. In *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing (HPDC '96)*, pages 551–561, Los Alamitos, California, Agosto de 1996. IEEE Computer Society Press.

[41] Michael Litzkow, Miron Livny e Matt Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104–111, Junho de 1988.

[42] MAG. `http://www.lsd.ufma.br/mag`. Último acesso em Julho/2005.

[43] MyGrid/OurGrid. `http://www.ourgrid.org`. Último acesso em Julho/2005.

[44] Klara Nahrstedt, Hao hua Chu e Srinivas Narayan. QoS-aware Resource Management for Distributed Multimedia Applications. *Journal of High-Speed Networks, Special Issue on Multimedia Networking*, 7(3,4):229–257, Março de 1999.

[45] Anand Natrajan, Michael Crowley, Nancyc Wilkins-Diehr, Marty Humphrey, Anthony D. Fox, Andrew S. Grimshaw e Charles L. Brooks III. Studying Protein Folding on the Grid: Experiences Using CHARMM on NPACI Resources under Legion. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC '10)*, pages 14–21. IEEE Computer Society, 2001.

[46] OiL. `http://oil.luaforge.net`. Último acesso em Julho/2005.

[47] Lawrence J. Hubert Phipps Arabie e Geert De Soete, editors. *Clustering and Classification*. World Scientific, 1996.

[48] Procps. the Procps FAQ: `http://procps.sourceforge.net/faq.html`. Último acesso em Julho/2005.

[49] Manuel Román, Fabio Kon e Roy Campbell. Reflective Middleware: From Your Desk to Your Hand. *IEEE Distributed Systems Online*, 2(5), Julho de 2001.

[50] SETI@home. `http://setiathome.ssl.berkeley.edu`. Último acesso em Julho/2005.

[51] V. S. Sunderam. PVM: a framework for parallel distributed computing. *Concurrency, Practice and Experience*, 2(4):315–340, 1990.

[52] Weiqin Tong, Jingbo Ding e Lizhi Cai. A Parallel Programming Environment on Grid. *Lecture Notes in Computer Science*, 2657:225–234, 2003.

[53] Weiqin Tong, Jingbo Ding e Lizhi Cai. Design and Implementation of a Grid-Enabled BSP. In *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003)*, 2003. Disponível em: `http://ccgrid2003.apgrid.org/online_posters/index.html`. Último acesso em Fevereiro/2005.

[54] Leslie G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[55] Derek Wright. Cheap cycles from the desktop to the dedicated cluster: combining opportunistic and dedicated scheduling with Condor. In *Proceedings of the Linux Clusters: The HPC Revolution Conference*, Champaign - Urbana, IL, Junho de 2001.

Herramienta de Minería de Consultas para el Diseño del Contenido y la Estructura de un Sitio Web

Bárbara Poblete
Profesor Guía: Dr. Ricardo Baeza-Yates
CIW - Centro de Investigación de la Web
Dpto. de Cs. de la Computación
Universidad de Chile
Santiago, Chile
{bpoblete,rbaeza}@dcc.uchile.cl

Abstract

This paper presents the design and implementation of a model for mining user queries, structure and contents within a website. The aim of this model is to discover valuable information for improving the contents and structure of the site. The improvements suggested by the model are basically: adding new contents to the website, extending the current coverage of topics that interest users and adding or changing words used to describe links between documents. Additionally this model provides a method of validating the structure given by the links in the website in relation to the organization of its contents. The prototype of this model generates as an output several reports which allow the site's administrator to visualize possible issues and ways to improve these problems.

Keywords: Web mining, website mining, queries, search engines, clustering, website improvement

Resumen

En este trabajo se desarrolló e implementó un modelo de análisis de consultas, estructura y contenido de un sitio Web. La principal motivación para este trabajo fue el descubrir información útil, a partir de las consultas formuladas por los usuarios de un sitio, que permitan realizar mejoras tanto en el contenido como en la estructura de éste. Las mejoras propuestas por el modelo consisten básicamente en agregar nuevos contenidos al sitio, ampliar la cobertura de ciertos temas de interés para los usuarios y agregar o cambiar las palabras utilizadas para describir los enlaces entre documentos. Además el modelo provee una forma de validar la estructura dada por los enlaces encontrados en un sitio con respecto a la organización del contenido al interior de éste. Para la implementación de este modelo se creó una herramienta de minería de consultas la cual genera como resultado numerosos informes, los que permiten visualizar los puntos conflictivos al interior del sitio y formas en que estos conflictos pueden ser mejorados por los administradores del sitio Web.

Palabras claves: minería Web, minería de sitios Web, consultas, buscadores, clustering, mejoramiento de sitios Web

1. Introducción

La Web se caracteriza por su acelerado crecimiento, su uso cada vez más masivo y el ser un medio altamente facilitador para los negocios. Esto ha generado la necesidad por parte de los proveedores de sitios Web, de hacer sitios más intuitivos y accesibles para los usuarios [28]. Es de vital importancia que los interesados en los contenidos de un sitio, sean capaces de encontrarlo en la Web y a su vez de recorrerlo de una forma que les resulte cómoda y fácil. El no tomar conciencia de este hecho puede significar la pérdida de muchos clientes. En este nuevo medio, los motores de búsqueda surgen como la aplicación por excelencia, ya que se han convertido en la herramienta más utilizada para alcanzar sitios en la red y conducir a los usuarios hacia la información que ellos buscan en un momento determinado.

Al navegar a través de la Web los usuarios dejan registro de todas sus acciones, principalmente en las bitácoras o *logs* de acceso de los servidores de los sitios y buscadores que visitan. Este registro refleja los patrones de navegación de los usuarios, las consultas que estos formularon en los motores de búsqueda internos de los sitios (en el caso que exista alguno) y además las consultas en buscadores globales que resultaron en requerimientos de documentos del servidor. Toda esta información es entregada por los usuarios en forma implícita y puede entregar la clave hacia los gustos e intereses de los usuarios en general y en forma particular para los diferentes sitios que estos recorren.

La mayoría de las consultas que son enviadas a un buscador y que están relacionadas a un sitio Web en particular, representan necesidades reales de los usuarios que visitan ese sitio. Sin embargo, las consultas de los usuarios sólo han sido estudiadas con el objetivo de mejorar motores de búsqueda y no con la intención de descubrir datos interesantes para mejorar la calidad de los contenidos y la estructura de los sitios Web. Es por este motivo que en este trabajo se presenta un modelo de minería Web que analiza las consultas al interior de un sitio, clasificándolas en diferentes categorías de acuerdo a su capacidad de contribuir a mejorar el sitio. Este modelo adicionalmente genera una visualización de la distribución de los contenidos del sitio en relación a la organización de los enlaces entre documentos, así como de las URLs seleccionadas como resultado a cada consulta. Los resultados generados por este modelo son presentados en diversos reportes desde los cuales es posible extraer recomendaciones para realizar mejoras al sitio Web.

Las contribuciones principales de este modelo para mejorar sitios Web son: *analizar las consultas de los usuarios al interior del sitio, obtener nuevos contenidos de interés para ampliar la cobertura de temas en el sitio, cambiar o agregar palabras en las descripciones de los hiper enlaces, agregar nuevos enlaces entre documentos relacionados y revisar los enlaces existentes entre documentos que no están relacionados entre sí por contenido.*

La implementación de este modelo ha sido aplicada en diferentes tipos de sitios, cada uno con diferentes volúmenes de tráfico y número de documentos. En todos los casos este modelo ha demostrado ser de gran utilidad al señalar formas de mejorar los sitios. Encontrándose que este modelo es especialmente útil en sitios con mucho contenido, los cuales se vuelven difíciles de organizar para sus administradores.

Este trabajo se organiza de la siguiente forma: en la sección 2 se muestra el trabajo relacionado y en la sección 3 se definen algunos conceptos preliminares. La sección 4 describe el modelo, mientras que en la sección 5 se muestra un breve resumen de un caso de uso de la herramienta y algunos de los resultados obtenidos. En la última sección se entregan algunas conclusiones y discute el trabajo futuro.

2. Trabajo Relacionado

La *minería Web* o *Web mining* [29] es el proceso de descubrir los diferentes patrones y relaciones al interior de un conjunto de datos de la Web. La minería Web puede ser dividida en tres tipos principales: la *minería de contenido*, la *minería de estructura* y la *minería de uso*. Cada uno de estos tipos de minería va unida principalmente al los siguientes tipos de datos encontrados en un sitio Web (ver figura 1):

Contenido: Son los datos reales que el sitio desea entregar sus usuarios, estos están compuestos generalmente de texto e imágenes. Este tipo de datos es el más importante y difícil de procesar, por ser multimedial.

Estructura: Son los datos que describen la organización del contenido al interior de un sitio. Esto incluye, la organización dentro de una página, de los enlaces, tanto internos al sitio como externos, y la jerarquía de todo el sitio (para más detalles ver [9]).

Uso: Son aquellos datos que describen el uso al cual se ve sometido un sitio, reflejado por los logs de acceso de los servidores Web.

Como se describe en [2] los datos obtenidos de la Web pueden ser analizados en forma dinámica o estática. Para el análisis estático se utiliza una *instantánea* de la Web en cierto momento y para estudiarla en forma dinámica se compara la evolución de la Web en instantes distintos de tiempo. Los datos además pueden ser obtenidos localmente, ya sea de un sitio en particular o en forma global, considerando una porción importante de la Web.

En el último tiempo la minería de uso ha generado un alto interés comercial [12, 3]. A partir del análisis de los logs de acceso se podría concluir, por ejemplo, que un documento que nunca es visitado no tiene razón de existir, o por el contrario, si una página muy visitada no se encuentra en los primeros niveles de jerarquía

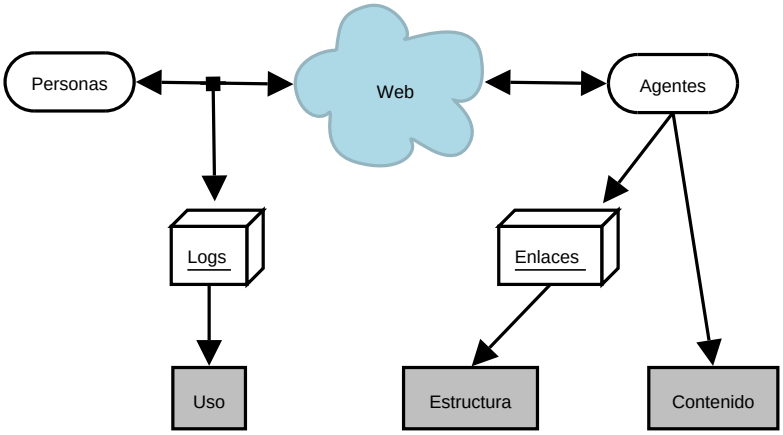


Figura 1: Tipos de datos en la Web.

de un sitio, esto sería un indicador para mejorar la organización y navegación. Los logs de tráfico de un sitio permiten, sin duda, hacer mejoras en la estructura y el contenido de estos para hacerlos más exitosos. Por lo tanto es importante detectar patrones de acceso y sus tendencias. Este tipo de análisis puede hacerse para un usuario en particular o en forma genérica.

En [28] se menciona que existen tres factores importantes que afectan la percepción de los usuarios de un sitio Web: el contenido, el diseño de las páginas y el diseño global del sitio. El primer factor mencionado corresponde a la información real que aporta el sitio y sus servicios. Los otros factores corresponden a la forma en que se hacen accesibles y comprensibles los contenidos en el sitio a los usuarios. Se hace la distinción entre el diseño de páginas individuales y el diseño global de un sitio, porque un sitio no es sólo una colección de páginas, sino que es una red de páginas relacionadas. Los usuarios no se tomarán el trabajo de explorar un sitio si este no posee una estructura intuitiva para ellos.

Existe una extensa lista de trabajos previos que utilizan la minería Web para mejorar sitios, la mayoría de los cuales se enfocan en apoyar el funcionamiento de sitios adaptativos [22] y la personalización automática de sitios [18]. Entre otras cosas, se analizan patrones frecuentes de navegación y reglas de asociación en base a las páginas visitadas por los usuarios, para descubrir reglas y patrones interesantes en un sitio [8, 28, 7, 10, 17]. Otras investigaciones se enfocan principalmente en el modelamiento de sesiones, perfiles de usuarios y análisis de clusters [14, 19, 20, 21, 23].

Las consultas formuladas en los motores de búsqueda pueden ser una herramienta valiosa para mejorar sitios Web y buscadores. La mayor parte del trabajo en esta área está dirigido al uso de las consultas para el mejoramiento de los buscadores internos a sitios Web [30] y a hacer más efectivos los buscadores de la Web global [6, 4, 15, 27]. Por otra parte, las consultas pueden ser estudiadas para mejorar la calidad de un sitio Web. Trabajo previo en este tema incluye el análisis de consultas similares en motores de búsqueda globales [13], lo cual permite encontrar consultas nuevas que son similares a las que ya conducen tráfico hacia un sitio, para posteriormente utilizarlas como información para mejorar el sitio. Otro tipo de análisis se presenta en [2] y consiste en estudiar las consultas formuladas en el buscador interno de un sitio, señalando que información valiosa puede ser encontrada a partir del análisis del comportamiento de los usuarios posterior a realizar una consulta. Este es el punto de partida a nuestro trabajo.

3. Conceptos Preliminares

En esta sección se presentan algunos conceptos importantes necesarios para entender el modelo de minería de consultas.

Logs de acceso: Son una parte vital para la minería Web, permiten descubrir, entre otras cosas, las necesidades de los visitantes con respecto al sitio que se está analizando. Los logs de acceso, registran en forma secuencial cada uno de los requerimientos vía HTTP al servidor Web. Es importante señalar que no todos los accesos a páginas de un sitio Web son registrados en los logs de acceso. Esto puede ocurrir

en el caso de que el usuario visite documentos almacenados en el *historial* de su navegador (o *browser*), lo cuál ocurre al utilizar los botones *volver (back)* o *avanzar (forward)*, en la barra de herramientas. En este caso los documentos desplegados por el cliente no son solicitados al servidor Web, sino que se extraen de una copia local que posee el usuario. Los campos más importantes que registra un log son los siguientes:

IP: Este campo denota la IP o nombre del host que realiza un requerimiento al servidor Web.

Time: En este campo se registra la fecha y hora en que se realiza el requerimiento.

Method: Este campo se refiere al método utilizado por el usuario para realizar su pedido, los métodos más frecuentes son `POST` y `GET`.

URL: En este campo se registra la URL del documento que es solicitado al servidor.

Referer: Este campo registra la URL desde la cual el usuario realiza el requerimiento del documento. En otras palabras, registra la página desde la cual se seleccionó el enlace que gatilló el requerimiento.

User Agent: Este campo registra el cliente Web que el usuario utilizó al momento de realizar su solicitud.

Sesión: Una sesión corresponde a todos los accesos registrados para un usuario, en los logs de uso de un sitio, dentro de un intervalo de tiempo máximo. Este intervalo es definido por defecto en 30 minutos, pero puede modificarse a cualquier otro valor adecuado para el sitio [11]. Cada usuario es identificado en forma única por su `IP` y `User-Agent` (esto es una heurística que puede ser mejorada [29]).

Motor de búsqueda: Un motor de búsqueda puede ser tanto interno como externo a un sitio Web. La principal diferencia entre un motor de búsqueda interno y uno externo, es que el motor de búsqueda interno sólo se remite a las páginas encontradas al interior de un sitio Web, mientras que el externo en general recopila los documentos de la Web en forma global.

Consultas: Una consulta corresponde a un conjunto de una o más palabras claves formuladas por un usuario en un motor de búsqueda, y representan una necesidad de información de ese usuario.

Ruta de Navegación: Es el camino a través de enlaces realizado por un usuario en una sesión, para alcanzar un documento determinado. Los documentos alcanzados de esta forma son considerados como *documentos navegados*.

Esencia de la Información: La esencia de la información [24] es un término que indica qué tan buena es una palabra con respecto a otras palabras con la misma semántica, siendo las palabras polisémicas (palabras con más de un significado) de menor esencia de la información. En general las consultas más frecuentes contienen mayor esencia de la información.

4. Descripción del Modelo

En esta sección presentamos una descripción del modelo de minería de contenido, estructura y uso en un sitio Web, enfocado en consultas. Los datos de entrada del modelo son: los contenidos del sitio, su estructura y sus logs de uso. La estructura del sitio es obtenida de los enlaces entre documentos y los contenidos corresponden al texto asociado a cada una de sus páginas. El objetivo de este modelo es generar información que permita mejorar la estructura y los contenidos de un sitio Web, además de evaluar la interconectividad entre documentos similares.

En la figura 3 se muestra una descripción del modelo, el cual recopila información de las consultas internas y externas del sitio, los patrones de navegación y los contenidos, para descubrir esencia de la información que se pueda utilizar para mejorar el contenido del sitio. Adicionalmente, los datos de los enlaces y contenidos del sitio son analizados usando *clustering* de documentos similares y componentes conexas. Estos procedimientos son explicados en mayor detalle a continuación.

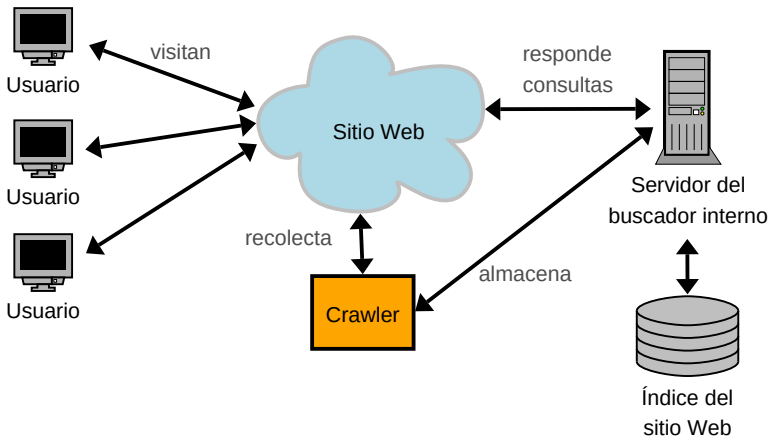


Figura 2: Motor de búsqueda interno.

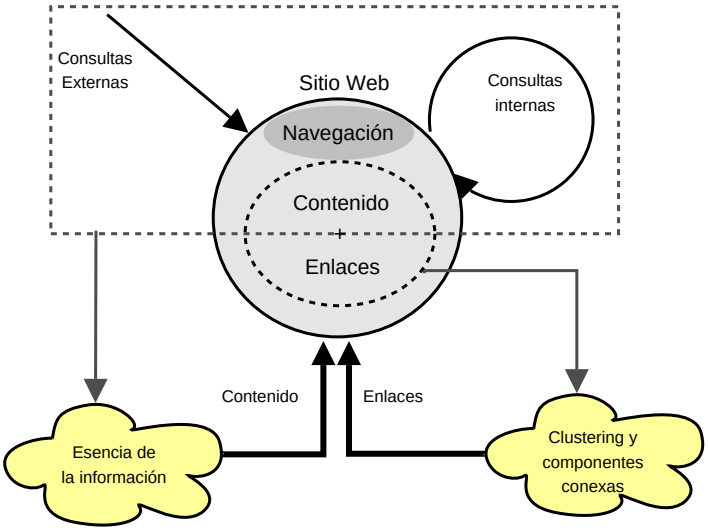


Figura 3: Modelo de la herramienta de minería de consultas.

4.1. Consultas

Generalmente en los logs de acceso a un sitio Web se pueden encontrar dos tipos de consultas (ver figura 3):

Consultas externas: Son las consultas realizadas a un buscador global (externo al sitio), a partir de las cuales los usuarios seleccionaron y visitaron los documentos de un sitio en particular. Estas consultas pueden ser descubiertas utilizando el campo `referer` de los logs de acceso y siempre son consultas satisfactorias para el usuario (son consideradas como satisfactorias debido a que el usuario visitó alguna respuesta).

Consultas internas: Son aquellas consultas formuladas en el buscador interno de un sitio. Adicionalmente, consultas en buscadores externos restringidas a un sitio en particular, son consideradas como consultas internas para ese sitio. Por ejemplo, consultas en Google.com que incluyen `site:example.com` son consultas internas para el sitio `example.com`. En este tipo de consultas, es posible encontrar consultas sin resultados visitados.

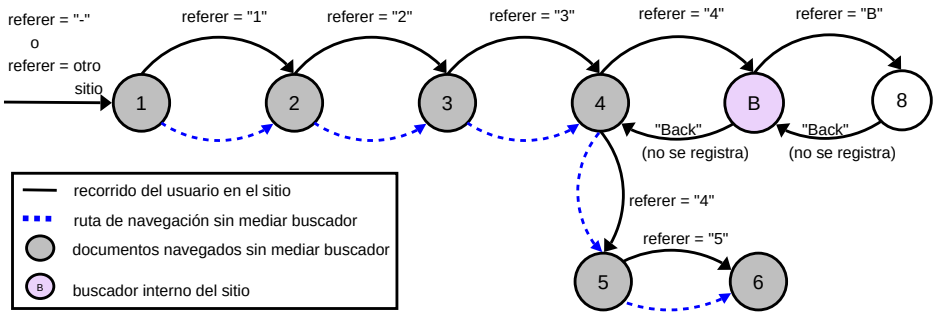


Figura 4: Documentos navegados sin mediar buscador.

Cada consulta formulada en un motor de búsqueda genera una página con enlaces a documentos, considerados por el buscador como resultados adecuados a la consulta. Una vez desplegada la página de resultados, el usuario puede visitar los documentos que le parezcan adecuados a su consulta. Esto gracias al *resumen* que acompaña cada uno de los enlaces en la página de resultados. El resumen muestra el contexto del documento en el cual aparece la consulta del usuario. Ésto permite a simple vista determinar los documentos que aparentemente son más adecuados según el usuario.

4.2. Modelo de Navegación

En el análisis realizado por este modelo son de especial interés tres conjuntos de documentos, los *documentos navegados sin mediar buscador*, *documentos alcanzados mediando el buscador interno* y *documentos alcanzados mediando un buscador externo*. A continuación se definen estos tipos de documentos:

Documentos Navegados Sin Mediar Buscador (DNSMB): Son aquellos documentos que, en el transcurso de una sesión, han sido alcanzados utilizando navegación a través de enlaces, sin utilizar un buscador interno u otros externos para encontrarlos. Es decir, los documentos alcanzados desde la página de resultados de un buscador y los documentos navegados desde estos resultados *no* son considerados en este conjunto. Sí son considerados en este conjunto, todos los documentos navegados desde documentos visitados antes de utilizar el buscador interno. De esta manera se admite la posibilidad de que el usuario realice una o varias búsquedas para luego volver a alguna de las páginas previas a esto y retomar la navegación. Este caso se ilustra en el diagrama de navegación presentado en la figura 4.

Documentos Alcanzados Mediando el Buscador Interno (DAMBI): Son aquellos documentos, que en el transcurso de una sesión, sólo son alcanzados por ser resultados directos de una consulta en un buscador interno del sitio Web. Estos documentos se deducen de los requerimientos que poseen `referer` igual a la URL de la página de resultados del buscador interno.

Documentos Alcanzados Mediando un Buscador Externo (DAMBE): Son aquellos documentos, que en el transcurso de una sesión, sólo son alcanzados por ser resultados directos de una consulta en un buscador externo al sitio Web. Estos documentos se deducen de los requerimientos que poseen `referer` igual a la URL de la página de resultados de un buscador externo.

Nótese que los conjuntos DNSMB, DAMBI y DAMBE *no son conjuntos disjuntos*, ya que pueden existir documentos que en alguna sesión sean alcanzados navegando sin pasar por un buscador y que en otra sean alcanzados sólo gracias a una consulta en un buscador. Lo importante es contabilizar el número de veces que estos diferentes eventos ocurren, ya que esto está directamente relacionado con la relevancia que tengan para el modelo.

En la figura 5 se muestra el diagrama de estados que representa a la heurística que fue necesario crear para clasificar los documentos DNSMB. Este método procesa secuencialmente todos los requerimientos producidos en una sesión y utiliza los campos el `referer` y la `URL` para decidir el siguiente estado. Nótese que `REFERER = consulta`, se refiere a que el requerimiento proviene de una consulta a un buscador interno o externo. El conjunto DNSMB puede ser inicialmente un conjunto de URLs correspondientes a las principales páginas de ingreso al sitio en estudio.

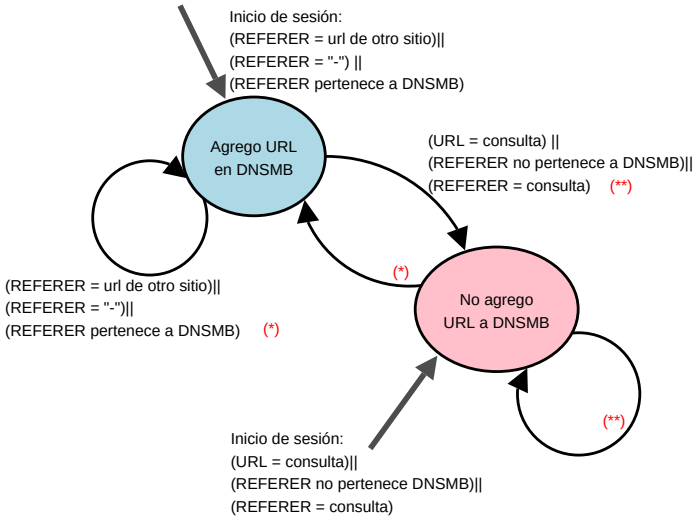


Figura 5: Heurística para decidir si un documento es DNSMB.

4.3. Clasificación de Consultas

4.3.1. Consultas Clase A y B

Una vez formulada una consulta, el usuario puede elegir visitar, uno o más de los resultados desplegados por el buscador. En el caso que el usuario visite uno o más resultados, se dirá que la consulta formulada corresponde a una consulta clase A o clase B. Este tipo de análisis es válido tanto para consultas internas como externas. A continuación se definirán estas dos clases de consultas (ilustradas en la figura 6):

Consulta clase A: Corresponde a consultas para las cuales los usuarios eligieron visitar uno o más documentos *DA*, donde DA contiene documentos en DNSMB. En otras palabras, si al formular la consulta el usuario visita algún documento que también ha sido alcanzado, en alguna sesión, navegando sin mediar buscador, se dice que esta consulta es de clase A.

Consulta clase B: Corresponde a consultas para las cuales los usuarios eligieron visitar uno o más documentos *DB*, donde DB contiene documentos que sólo pertenecen a DAMBI o DAMBE y no a DNSMB. Es decir, los documentos en DB sólo han sido alcanzados utilizando algún tipo de buscador pero nunca por medio de una ruta de navegación en la cual no intervenga un buscador.

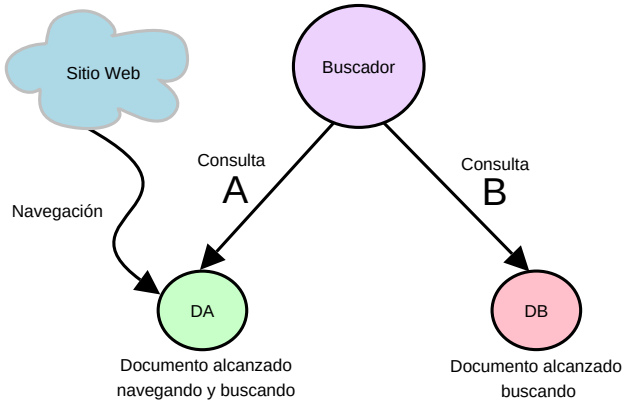


Figura 6: Clases de consultas A y B.

La finalidad de definir estas dos clases de consultas es que, las consultas de clases A y B permitirían encontrar nuevas palabras para describir los enlaces a documentos contenidos en DA y DB. En el caso de las consultas clase A, estas contienen palabras igual de buenas que las existentes para describir los enlaces a documentos en DA, aportando *esencia de la información adicional a la existente*. En el caso de las consultas clase B, estas contienen conceptos completamente nuevos para utilizar en las descripciones de los enlaces a DB, ya que en este caso, las palabras utilizadas para describir los enlaces a DB no son asociadas por los usuarios con estos documentos. Las consultas clase B representarían entonces *nueva esencia de la información para el sitio*.

4.3.2. Consultas Clase C, C' y D

El otro caso que se presenta al formular una consulta interna, es aquel en que, de los resultados desplegados por el buscador, el usuario no elija ninguno para visitar. Esto puede ocurrir principalmente a dos motivos:

1. Que el buscador no haya desplegado resultados.
2. Que ninguno de los resultados presentados correspondiera con el concepto que buscaba el usuario, a pesar de que estos contienen las palabras de la consulta. Esto ocurre en el caso de consultas con varios significados.

Para clasificar estas consultas es necesario separar las diferentes causantes. Para esto se crearon las clases C, C' y D, que se describen a continuación (ver figura 7):

Consulta clase C: Corresponde a consultas para las cuales el buscador desplegó uno o más resultados, pero que sin embargo el usuario no eligió documentos para visitar.

Consulta clase C': Corresponde a consultas para las cuales el buscador desplegó cero documentos como resultado, debido a que las palabras utilizadas en la consulta no existen dentro del sitio. Luego, al realizar una clasificación manual el administrador del sitio determina que el concepto representado por la consulta *existe* en el sitio pero explicado con otras palabras.

Consulta clase D: Corresponde a consultas para las cuales el buscador desplegó cero documentos como resultado, debido a que las palabras utilizadas en la consulta no existen dentro del sitio. Luego, al realizar una clasificación manual el administrador del sitio determina que el concepto representado por la consulta *no existe* en el sitio.

Esta división en clases C, C' y D, es importante ya que cada caso aporta diferentes sugerencias para el sitio. Las consultas clase C representan conceptos que existen al interior del sitio, pero con un significado diferente al que buscan los usuarios. Este tipo de consultas sugiere *agregar contenido al sitio*, pero con el sentido que solicitan los usuarios. Las consultas de clase C', corresponden a conceptos que existen en el sitio pero descritos con otras palabras. Por lo tanto sugieren *agregar los términos de las consultas clase C' en los documentos que corresponden a estos conceptos dentro del sitio*. Además de utilizar estas nuevos términos para describir los enlaces que conducen a estos documentos. Las consultas clase D, corresponden a conceptos que no existen en el sitio, pero que dada la frecuencia con que son buscados por los usuarios, *proponen nuevos contenidos para el sitio o nuevos elementos de valor comercial para éste*.

Es importante señalar que la clasificación de las consultas C' y D son almacenadas en un *historial* para poder agilizar posteriores procesos de clasificación manual.

4.4. Tesauro

Este modelo incluye un tesauro simple, que agrupa las consultas que poseen un mismo significado. Este tesauro es creado utilizando el *feedback* entregado por el usuario de la herramienta, él cual debe asociar manualmente las consultas correspondientes. Una vez realizado el agrupamiento de consultas de idéntico significado, los resultados desplegados por el modelo son agrupados según los términos presentes en el tesauro.

En régimen permanente el tesauro no requiere de mucha clasificación manual, ya que los términos que contiene son guardados en forma permanente para poder ser reutilizados en posteriores procesos de minería en el sitio Web.

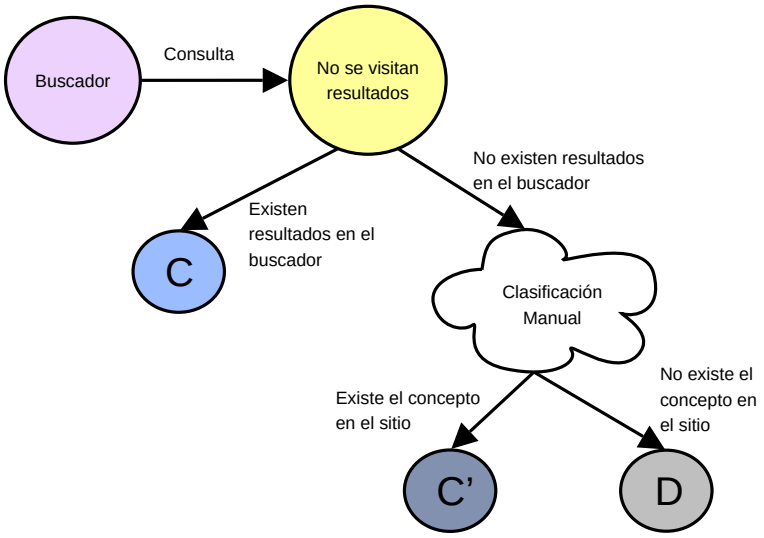


Figura 7: Tipos de consultas C, C' y D.

4.5. Clustering

El *clustering* es una técnica que consiste en agrupar elementos en conjuntos (o *clusters*) de similares características, de manera que los elementos al interior de un conjunto tengan mayor similaridad entre sí, que con elementos de otros conjuntos. El número de conjuntos que se utiliza es un parámetro que se define al momento de realizar el clustering.

Es importante mencionar que para realizar clustering existen diversos métodos posibles. Pero en este caso se utiliza el enfoque de *bisecciones secuenciales*, que consiste en comenzar con un solo cluster con todos los elementos en su interior y dividirlo en dos conjuntos, luego nuevamente se elige uno de los conjuntos resultantes para dividirlo en dos, y así sucesivamente hasta lograr el número de clusters deseado. El cluster elegido, para ser dividido cada vez, es aquel que maximice la función global de clustering de la ecuación 1.

$$\sum_{i=1}^{k} \sqrt{\sum_{v,u \in S_i} sim(u,v)} \tag{1}$$

En donde k es el número total de clusters, S_i es el número de elementos del cluster i-ésimo, v, u representan dos objetos y $sim(u,v)$ corresponde a la similaridad entre los dos objetos definida por la función coseno. Esta función se considero la más adecuada para este proceso y es discutida con mayor detalle en [16].

En este modelo el clustering es aplicado al texto de los documentos del sitio Web, en donde cada documento es representado internamente como un vector cuyas coordenadas corresponden a la frecuencia con que aparece cada palabra del vocabulario del sitio. Este procedimiento tiene como objetivo encontrar conjuntos de documentos de contenidos similares. El objetivo de esto es verificar la interconectividad (por medio de enlaces) entre conjuntos similares de documentos.

Es importante mencionar que *al realizar clustering utilizando bisecciones secuenciales se crea un árbol jerárquico de clusters, llamado dendograma*. Este tiene en su raíz el conjunto inicial, el cual contiene a todos los elementos. Al realizarse las bisecciones se comienza a ramificar en conjuntos más pequeños hasta que el árbol contiene tantas hojas como clusters deseados. Al analizar este árbol se puede observar cuáles clusters (hojas) son más similares que otros, ya que descienden del mismo nodo padre y cuáles son más distintos, porque las ramas del árbol a la cual corresponden se unen en la parte superior de la jerarquía. En la figura 8 se muestra un ejemplo de un árbol jerárquico para un proceso de clustering de 7 clusters. En este ejemplo se puede apreciar que los clusters en color azul son similares entre sí, ya que fueron separados del mismo nodo padre, al igual que los clusters rojos entre ellos. Sin embargo, los clusters azules y los rojos no se parecen mucho entre ellos, ya que fueron separados en en la etapa más temprana de bisección.

Este modelo analiza los nodos de cada nivel del árbol jerárquico, estudiando el número de enlaces entre

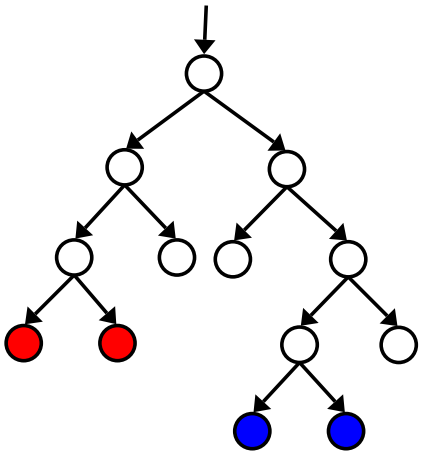


Figura 8: Árbol jerárquico para 7 clusters.

documentos pertenecientes a nodos de un mismo nivel, con el objetivo de validar la interconectividad de los diferentes nodos según la similaridad que estos presentan en el árbol jerárquico. Como resultado se genera una visualización en base a colores para indicar entre cuales clusters existen problemas de interconectividad según el número de enlaces entre nodos de un mismo nivel de jerarquía.

4.6. Arquitectura de la Herramienta

En la figura 9 se muestra la arquitectura de la herramienta de minería de consultas Web y los agentes externos que se ven involucrados en este proceso. A continuación se describe la arquitectura y las diferentes etapas enumeradas en ésta:

1. El sitio se encuentra en la Web y su funcionamiento está dado principalmente por dos servidores: el servidor Web, que es el encargado de responder los requerimientos HTTP, y el servidor del buscador interno del sitio, que es el encargado de responder las consultas que llegan desde el sitio Web. El sitio recibe pedidos de páginas y otros contenidos, de los usuarios que lo visitan y navegan a través de él.

2. El sitio es recolectado por un *crawler* o robot que se encarga de recorrer en forma íntegra sus contenidos y recopilar todos los documentos, que se encuentran disponibles en la Web de éste, junto con la información de los enlaces que los unen. Estos datos son almacenados en el servidor del buscador interno.

3. Todos los requerimientos realizados por los usuarios son registrados en los logs de acceso del servidor Web y del buscador interno. Estos logs posteriormente se unen, se limpian de información irrelevante y se extraen las sesiones a un archivo de sesiones.

4. El archivo de sesiones pasa por un proceso de análisis en el cual se extrae la información que se considera útil respecto de las consultas, documentos y estadísticas generales. Posteriormente esta información compara y clasifica las diferentes clases de consultas encontradas, de acuerdo al impacto que tendrían en la estructura y el diseño del sitio.

5. El log de acceso al buscador es procesado en busca de patrones de consulta frecuentes. Es decir, se buscan conjuntos de palabras que se repitan frecuentemente en las consultas.

6. Los documentos de la colección de documentos del sitio Web, son almacenados en el servidor del buscador, son indexados y los archivos que componen el índice son procesados para separar la estructura y el contenido.

7. Se realiza clustering sobre el contenido agrupando los documentos en los diferentes clusters, construyéndose el árbol jerárquico de este proceso. Los datos obtenidos del clustering son comparados con

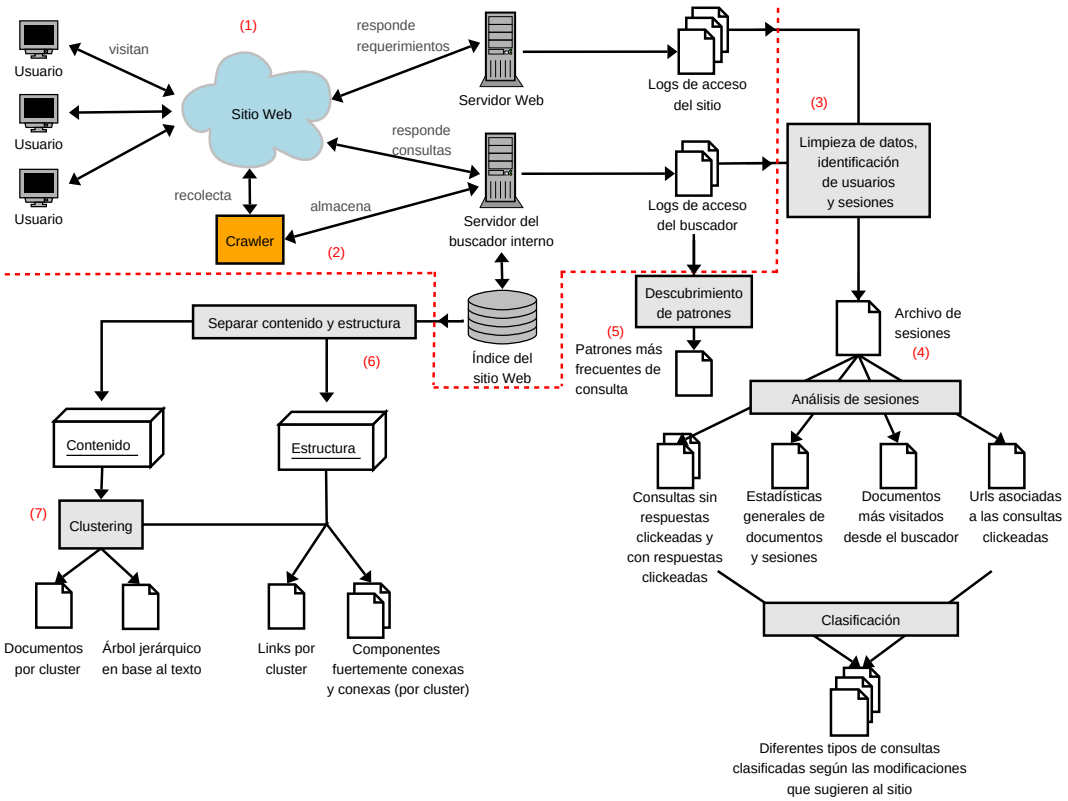


Figura 9: Arquitectura de la herramienta de minería de consultas.

la información de los enlaces dada por la estructura, para así obtener información de qué tan bien se encuentra relacionada la estructura del sitio con su contenido.

La línea punteada en la figura 9 marca la separación entre las componentes propias del sitio Web y la herramienta de minería de consultas.

5. Prototipo y Caso de Uso

La implementación del modelo involucró diferentes tareas de minería Web, presentadas en la sección anterior. Los datos de los contenidos y la estructura se extraen del índice generado por el recolector de páginas del motor de búsqueda del sitio [1], la herramienta utilizada para el clustering de texto es CLUTO [16] y este proceso utiliza el método de bisecciones secuenciales, pero nuestro modelo puede incorporar cualquier otra técnica de clustering. Para el análisis de patrones de consulta se utilizó LPMINER [26]. Las consultas internas provienen del buscador interno y consideramos sólo consultas externas provenientes de Google (aunque pueden incorporarse más buscadores externos). Más detalles de la implementación en [25].

Los informes generados por la herramienta son bastante detallados y requieren de un análisis detenido por parte del administrador de cada sitio. Para ejemplificar este caso se presenta una muestra de los resultados principales y más descriptivos obtenidos a partir de un caso de uso real.

Para este caso de uso se analizó un sitio, al que llamaremos *sitio A*, que corresponde a un portal de alto tráfico en Internet, enfocado principalmente a universitarios y alumnos de la enseñanza superior. Abarca una amplia gama de contenidos diferentes, desde contenidos culturales, pasatiempos, noticias universitarias, estudios y universidades entre muchos otros. Este sitio se encuentra compuesto de aproximadamente unos 4100 documentos en su sitio principal, aunque además posee otros subsitios que no son considerados en este análisis por abarcar contenidos diferentes. Los logs de este sitio registran un volumen diario de requerimientos de 180 Mb aproximadamente.

Fecha de inicio de logs:	22/Sep/2004
Fecha de término de logs:	28/Sep/2004
Número total de sesiones:	56.000
Promedio páginas vistas por sesión:	10,05
Número de páginas visitadas (contadas 1 vez por sesión):	360.000
Número de consultas internas:	4.500
Número de consultas externas:	19.300
Número de respuestas clickeadas:	20.000

Tabla 1: Información aproximada de uso del *sitio A*.

Porcentaje	**Patrón**
1,698 %	cursos
1,698 %	ensayos psu
1,201 %	aranceles
0,953 %	uniacc
0,829 %	becas
0,787 %	simulador *(Existe sólo 1 respuesta)*
0,746 %	credito universitario *(Existe sólo 1 respuesta)*
0,621 %	kinesiologia
0,621 %	tesis
0,580 %	magister *(Existen 3 respuestas)*
0,580 %	curriculum
0,580 %	psu puntajes *(Existen 2 respuestas)*
0,539 %	diplomados *(Existen 0 respuestas)*
0,539 %	traductor
0,539 %	seminarios *(Existen 4 respuestas)*
0,539 %	antofagasta
0,497 %	americas
0,497 %	publicidad
0,497 %	chilenas universidades
0,497 %	civil industrial ingenieria
0,456 %	psicologia
0,456 %	modelos
0,456 %	test vocacional *(Existen 0 respuestas)*

Tabla 2: Patrones más frecuentes en las consultas internas del *sitio A*.

En la tabla 1 se presenta la información general del análisis de los logs de uso del *sitio A*, en este caso se analizó un log equivalente a una semana.

La tabla 2 refleja los patrones más frecuentes encontrados en las consultas internas del sitio. En este caso se puede apreciar que los intereses de los usuarios claramente se encuentran dirigidos a temas universitarios, tales como *"cursos"*, *"ensayos de la psu*[1] *"*, *"aranceles"*, *"becas"*, *"carreras"* e *"institutos"*, entre muchos otros. En esta tabla destacan consultas como *"simulador"*, *"credito universitario"*, *"psu puntajes"* y *"magister"*, para las cuales existen pocos documentos apropiados al interior del sitio. De los patrones más frecuentes indican cuales temas deberían ser alcanzables desde las páginas principales del sitio. Además las consultas con pocas respuestas merecen una revisión por parte del editor del sitio, para estudiar la posibilidad tratar estos temas con mayor profundidad en el contenido, dado que reflejan los intereses de los usuarios.

En la figura 10 se muestran los documentos más visitados en el *sitio A* mostrados desde la interfaz de la herramienta, como resultado de una consulta en el buscador interno. De esta tabla se obtiene información sobre los documentos que son más populares en el buscador interno, estos documentos en general deberían ser accesibles desde las páginas principales del sitio. Adicionalmente se entrega información sobre el cluster

[1]PSU: Prueba de Selección Universitaria en Chile.

Número de enlaces internos	1.677
Número de documentos en el cluster	202
Número de componentes fuertemente conexas	41
Número de componentes conexas	8
Promedio similaridad interna	0,047
Promedio similaridad externa	0,014

Tabla 3: Datos del cluster 29 del *sitio A*.

al cual pertenece cada documento y el tamaño de este cluster, lo cuál permite ver si existen en el sitio otros documentos similares a estos. Para cada documento en la tabla es posible apreciar si pertenece a un cluster con buena interconectividad, lo cual es deseable ya que permite a los usuarios que visitan estos documentos navegar hacia documentos de contenidos similares. Por ejemplo en la figura 10 el documento más frecuente corresponde al cluster 29. Al revisar la información de este cluster en la tabla 3, se puede apreciar que es un cluster cuyos documentos se encuentran relativamente bien conectados entre sí, ya que posee numerosos enlaces internos y pocas componentes conexas.

En la figura 11, se presentan las consultas internas más frecuentes de clase A, estas consultas aportan esencia de la información adicional a la que ya existe al interior del sitio, para describir los documentos asociados a cada consulta. Es decir, es una buena idea incorporar las palabras definidas en las consultas clase A a los textos utilizados en los enlaces dirigidos los documentos asociados. Por ejemplo las palabras "ensayos psu" son buenas para describir los enlaces al documento http://www.sitioA.cl/mde/Ensayos.jsp.

Las consultas internas clase B, que se muestran en la figura 12, condujeron a los usuarios hacia documentos que normalmente no suelen ser alcanzados sin utilizar un buscador interno o externo. Por lo tanto estas consultas sugieren *nueva esencia de la información* para los enlaces dirigidos a los documentos asociados a estas consultas. Por medio de estas consultas es posible detectar documentos de interés para los usuarios en temas como *"ensayos psu"*, *"facsimil"*[2], *"embajadas"* y *"universidades"* que no son alcanzados por la mayor parte de los usuarios al navegar a través del sitio.

Las consultas clase C, mostradas en la tabla 4, representan las consultas para las cuales el buscador interno arrojó respuestas, pero sin embargo los usuarios decidieron no visitar ninguna de estas. En estos casos lo recomendable es mejorar la cobertura de los temas representados por estas consultas ya que para algunos usuarios del sitio esta no es adecuada. Estos casos corresponden a conceptos como *"becas"*, *"psu"*, *"carreras"* y *"chile universidad"*, entre otros.

En la tabla 5 se presentan las consultas de tipo C' encontradas a partir de la clasificación manual de las consultas sin respuestas visitadas. En este informe se pueden apreciar conceptos que existen al interior del *sitio A* pero descritos con otras palabras. En este caso lo recomendable es incorporar estas palabras en los documentos que tratan estos temas y a su vez utilizar estas palabras para describir enlaces hacia estos documentos.

Las consultas clase D, presentadas en la tabla 6, corresponden a temas que los usuarios buscan al interior del sitio pero que no existen dentro de éste. Las consultas clase D, en este caso en particular, son de muy baja ocurrencia y no se considera de gran importancia revisar estos temas ya que no parecen pertinentes al sitio, pero en caso de presentarse consultas de mayor frecuencia de este tipo, lo recomendable sería analizar la incorporación de estos temas a los contenidos del *sitio A*.

La siguiente etapa en este análisis es revisar los datos generados por la herramienta específicos a la estructura y contenido del sitio. El informe de *análisis de clusters de texto y enlaces del sitio A*, mostrado en la figura 13, permite visualizar qué tan bien interconectado es el *sitio A*. En este caso se puede apreciar una alta ocurrencia de celdas color naranjo, lo que refleja que existe una gran cantidad de enlaces entre documentos de baja similaridad, como es el caso de los 1.656 enlaces desde el nodo 36 al nodo 29. Al mirar el detalle de este caso utilizando los reportes, se puede comprobar que todos los enlaces son un número fijo para cada documento, lo que corresponde a menús de navegación que se repiten en cada página. Esto no representa un problema en la organización de los enlaces.

Otro caso interesante de analizar son las celdas en color violeta y gris, que corresponden a nodos que a pesar de estar compuestos de documentos similares no poseen ningún enlace entre ellos. Un ejemplo de esto es el nodo 8, que como se puede apreciar en el *Nivel 5* del informe 13, no posee enlaces internos.

[2]Se debe recordar que los acentos fueron eliminados para este análisis.

Documentos más visitados desde el buscador interno
http://www.sitioA.cl

url	frec	cluster	tamaño	consultas
http://www.sitioA.cl/mde/Ensayos.jsp	49	29	203	ensayos *(19 clicks)* ensayos psu *(19 clicks)* ejercicios *(6 clicks)* lenguaje psu *(1 clicks)* ejercicios fisica resueltos *(1 clicks)* pedro preuniversitario valdivia *(1 clicks)* psu *(1 clicks)* demre *(1 clicks)*
http://www.sitioA.cl/servicios/buscadores/buscadores/inde...	30	--	0	fisica instituto matematica talca universida *(3 clicks)* adolfo ibañez *(2 clicks)* industrial ingenieria *(2 clicks)* corfo *(1 clicks)* univercidad valparaiso *(1 clicks)* humanos recursos *(1 clicks)* estadisticas *(1 clicks)* psicologia *(1 clicks)* turf *(1 clicks)* administraciòn producciò *(1 clicks)* hizmeri *(1 clicks)* ezequiel sponton *(1 clicks)* universidadvalparaiso *(1 clicks)* chileno *(1 clicks)* contruccion simbolos *(1 clicks)* becas españa grado post *(1 clicks)* ciencias politicas *(1 clicks)* nem *(1 clicks)* administracion ejecucion ingenieria *(1 clicks)* medicina *(1 clicks)* beca gomez milla postula *(1 clicks)* carreras estresantes *(1 clicks)* usach *(1 clicks)* binomios *(1 clicks)* talca universidad *(1 clicks)* antropologia docotorado *(1 clicks)*
http://www.sitioA.cl/includes/blocks/validaLoginHome.jsp	12	29	203	ensayos psu *(7 clicks)* psu *(3 clicks)* chilenas universidades *(2 clicks)*
http://www.sitioA.cl/servicios/buscador/carreras.php?area...	11	28	216	buscador carreras *(4 clicks)* dietetica nutricion *(4 clicks)* odontologia *(2 clicks)* ocupacional terapia *(1 clicks)*
http://www.sitioA.cl/servicios/conexionsitioA/como_inc...	8	--	0	ensayos psu *(5 clicks)* pruebas psu *(1 clicks)* ensayos *(1 clicks)* psu *(1 clicks)*
http://www.sitioA.cl/servicios/buscador/carreras.php?area...	8	28	216	carreras *(6 clicks)* buscador carreras *(2 clicks)*
http://www.sitioA.cl/servicios/buscador/descripcion_carre...	7	28	216	sociologia *(7 clicks)*
http://www.sitioA.cl/servicios/buscador/descripcion_carre...	7	28	216	veterinaria *(6 clicks)* medicina *(1 clicks)*
http://www.sitioA.cl/servicios/buscador/descripcion_carre...	7	28	216	odontologia *(5 clicks)* odontologìa *(2 clicks)*
http://www.sitioA.cl/servicios/buscador/descripcion_carre...	6	28	216	enfermeria *(6 clicks)*
http://www.sitioA.cl/contenidos/centros/centroseinstituto...	6	28	216	instituto profesional santiago *(2 clicks)* institutos *(2 clicks)* catolica temuco universidad *(1 clicks)* arturo prat universidad *(1 clicks)*
http://www.sitioA.cl/servicios/buscador/carreras.php?area...	6	28	216	carreras *(3 clicks)* publicas relaciones *(3 clicks)*
http://www.sitioA.cl/servicios/buscador	5	29	203	simulador *(5 clicks)*
http://www.sitioA.cl/servicios/buscador/descripcion_carre...	5	28	216	vespertino *(5 clicks)*
http://www.sitioA.cl/servicios/chat	5	--	0	ensayos *(2 clicks)* arquitectonico constructivismo *(1 clicks)* algebra *(1 clicks)* becas *(1 clicks)*
http://www.sitioA.cl/contenidos/alojamiento/alojamiento.h...	5	29	203	barrio universitario *(1 clicks)* residencia *(1 clicks)* pensiones *(1 clicks)* capital sociedad *(1 clicks)* residencias universitarias *(1 clicks)*
http://www.sitioA.cl/contenidos/orientacion/psu/calendari...	5	--	0	facsimil *(2 clicks)* admision requisitos *(1 clicks)* ensayo psu *(1 clicks)* ensayo psu resultados *(1 clicks)*
http://www.sitioA.cl/servicios/buscador/descripcion_carre...	5	28	216	kinesiologia *(5 clicks)*
http://www.sitioA.cl/concurso/p_incognito/interior.jsp	5	29	203	ensayos *(5 clicks)*
http://www.sitioA.cl/contenidos/orientacion/orientacion/o...	5	--	0	test vocacional *(4 clicks)* orientacion test vocacional *(1 clicks)*
http://www.sitioA.cl/servicios/buscador/ficha.php?carrera...	4	26	122	dietetica nutricion *(4 clicks)*
http://www.sitioA.cl/servicios/buscador/descripcion_carre...	4	15	47	biotecnologia *(4 clicks)*
http://www.sitioA.cl/servicios/buscador/descripcion_carre...	4	15	47	telecomunicaciones *(4 clicks)*

Figura 10: Documentos más visitados a partir de consultas internas del *sitio A*.

Clase A
http://www.sitioA.cl

similar escencia de la info. (sugerencias para enlaces)	documentos asociados
ensayos psu*(frecuencia: 24)*	http://www.sitioA.cl/mde/Ensayos.jsp *(19 clicks)* http://www.sitioA.cl/includes/blocks/validaLoginHome.jsp *(7 clicks)* http://www.sitioA.cl/servicios/conexionsitioA/como_inc... *(5 clicks)* http://www.sitioA.cl/includes/blocks/login.jsp *(2 clicks)* http://www.sitioA.cl/servicios/sms *(1 clicks)* http://www.sitioA.cl/index.htm *(1 clicks)* http://www.sitioA.cl/concurso/p_incognito/gracias.jsp *(1 clicks)* http://www.sitioA.cl/portada/actualidad/masactualidad.jsp *(1 clicks)*
ensayos*(frecuencia: 17)*	http://www.sitioA.cl/mde/Ensayos.jsp *(19 clicks)* http://www.sitioA.cl/concurso/p_incognito/interior.jsp *(5 clicks)* http://www.sitioA.cl/servicios/chat *(2 clicks)* http://www.sitioA.cl/servicios/conexionsitioA/como_inc... *(1 clicks)*
becas*(frecuencia: 8)*	http://www.sitioA.cl *(4 clicks)* http://www.sitioA.cl/contenidos/estudiantes/Alumnos.htm *(1 clicks)* http://www.sitioA.cl/concurso/p_incognito/interior2.jsp *(1 clicks)* http://www.sitioA.cl/concurso/p_incognito/gracias.jsp *(1 clicks)* http://www.sitioA.cl/servicios/chat *(1 clicks)*
carreras*(frecuencia: 7)*	http://www.sitioA.cl/servicios/buscador/carreras.php?area... *(6 clicks)* http://www.sitioA.cl/servicios/buscador/carreras.php?area... *(3 clicks)* http://www.sitioA.cl *(1 clicks)* http://www.sitioA.cl/servicios/buscador/descripcion_carre... *(1 clicks)*
psu*(frecuencia: 7)*	http://www.sitioA.cl/includes/blocks/validaLoginHome.jsp *(3 clicks)* http://www.sitioA.cl/contenidos/orientacion/psu/psu.htm *(2 clicks)* http://www.sitioA.cl/registro/olvidado *(1 clicks)* http://www.sitioA.cl/servicios/conexionsitioA/como_inc... *(1 clicks)* http://www.sitioA.cl *(1 clicks)* http://www.sitioA.cl/includes/blocks/login.jsp *(1 clicks)* http://www.sitioA.cl/concurso/p_incognito/gracias.jsp *(1 clicks)* http://www.sitioA.cl/mde/Ensayos.jsp *(1 clicks)*
test vocacional*(frecuencia: 5)*	http://www.sitioA.cl/contenidos/orientacion/orientacion/o... *(4 clicks)* http://www.sitioA.cl *(1 clicks)*
odontologia*(frecuencia: 4)*	http://www.sitioA.cl/servicios/buscador/descripcion_carre... *(5 clicks)* http://www.sitioA.cl/servicios/buscador/carreras.php?area... *(2 clicks)* http://www.sitioA.cl/servicios/buscador/ficha.php?carrera... *(1 clicks)*
ensayo psu*(frecuencia: 4)*	http://www.sitioA.cl *(1 clicks)* http://www.sitioA.cl/concurso/psu/bases.html *(1 clicks)* http://www.sitioA.cl/contenidos/internet/paginas_contenid... *(1 clicks)* http://www.sitioA.cl/cgi-bin/e-clasificados_zona/classifi... *(1 clicks)*
facsimil*(frecuencia: 4)*	http://www.sitioA.cl/contenidos/orientacion/psu/publicaci... *(3 clicks)*
medicina*(frecuencia: 3)*	http://www.sitioA.cl *(1 clicks)* http://www.sitioA.cl/servicios/buscadores/buscadores/inde... *(1 clicks)* http://www.sitioA.cl/servicios/buscador/descripcion_carre... *(1 clicks)*
embajadas*(frecuencia: 3)*	http://www.sitioA.cl/contenidos/internacionales/Internaci... *(2 clicks)*
enfermeria*(frecuencia: 3)*	http://www.sitioA.cl/servicios/buscador/descripcion_carre... *(6 clicks)*
vespertino*(frecuencia: 3)*	http://www.sitioA.cl/servicios/buscador/descripcion_carre... *(5 clicks)* http://www.sitioA.cl/servicios/buscador/lista_carreras.ph... *(3 clicks)* http://www.sitioA.cl/servicios/buscador/lista_carreras.ph... *(3 clicks)* http://www.sitioA.cl/servicios/buscador/lista_carreras.ph... *(2 clicks)* http://www.sitioA.cl/servicios/buscador/descripcion_carre... *(2 clicks)* http://www.sitioA.cl *(1 clicks)* http://www.sitioA.cl/servicios/buscador/descripcion_carre... *(1 clicks)* http://www.sitioA.cl/servicios/buscador/ficha.php?carrera... *(1 clicks)* http://www.sitioA.cl/servicios/buscador/ficha.php?carrera... *(1 clicks)* http://www.sitioA.cl/servicios/buscador/descripcion_carre... *(1 clicks)* http://www.sitioA.cl/servicios/buscador/lista_carreras.ph... *(1 clicks)* http://www.sitioA.cl/servicios/buscador/lista_carreras.ph... *(1 clicks)* http://www.sitioA.cl/servicios/buscador/descripcion_carre... *(1 clicks)*
psu resultados*(frecuencia: 3)*	http://www.sitioA.cl/servicios/novedades/novedades_tres.h... *(3 clicks)* http://www.sitioA.cl/concurso/psu/bases.html *(1 clicks)*
universidades*(frecuencia: 3)*	http://www.sitioA.cl/contenidos/orientacion/que-estudiar/... *(1 clicks)* http://www.sitioA.cl/contenidos/universidades/Universidad... *(1 clicks)* http://www.sitioA.cl/contenidos/estudios/Estudios.htm *(1 clicks)*

Figura 11: Consultas clase A internas en el *sitio A*.

Clase B
http://www.sitioA.cl

Nueva escencia de la informacion	documentos asociados
ensayos psu *(frecuencia: 24)*	http://www.sitioA.cl/contenidos/orientacion/psu/hcs2_psu.... *(1 clicks)*
ensayo psu *(frecuencia: 4)*	http://www.sitioA.cl/contenidos/orientacion/psu/calendari... *(1 clicks)*
facsimil *(frecuencia: 4)*	http://www.sitioA.cl/contenidos/orientacion/psu/calendari... *(2 clicks)*
embajadas *(frecuencia: 3)*	http://www.sitioA.cl/contenidos/internacionales/estudiant... *(2 clicks)*
universidades *(frecuencia: 3)*	http://www.sitioA.cl/contenidos/universidades/Universidad... *(1 clicks)* http://www.sitioA.cl/contenidos/universidades/historia/Un... *(1 clicks)*
institutos *(frecuencia: 2)*	http://www.sitioA.cl/contenidos/centros/centroseinstituto... *(2 clicks)*
residencias universitarias *(frecuencia: 2)*	http://www.sitioA.cl/contenidos/alojamiento/alojamientodo... *(1 clicks)* http://www.sitioA.cl/contenidos/alojamiento/alojamientotr... *(1 clicks)*
contrato modelos *(frecuencia: 2)*	http://www.sitioA.cl/contenidos/empresa/contratos.htm *(1 clicks)*
cinetica energia *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/orientacion/psu/fis2_psu.... *(1 clicks)* http://www.sitioA.cl/contenidos/orientacion/psu/qui2_psu.... *(1 clicks)*
ciclo cine maria santa *(frecuencia: 1)*	http://www.sitioA.cl/entreten/sinniuno/200407/11.act *(1 clicks)*
carrete valdivia *(frecuencia: 1)*	http://www.sitioA.cl/entreten/35mm/index.act?fecha=100118... *(1 clicks)*
becas tradicionales universidades *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/universidades/Universidad... *(1 clicks)*
desarrollo fondo institucional *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/universidades/Universidad... *(1 clicks)*
ensayo psu resultados *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/orientacion/psu/calendari... *(1 clicks)*
privadas universidades *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/universidades/Universidad... *(1 clicks)*
uniacc *(frecuencia: 1)*	http://www.sitioA.cl/entreten/sinniuno/200408/8.act *(1 clicks)*
españa malaga universidades *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/estudiantes/centrojuvenil... *(1 clicks)*
iglesia *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/orientacion/psu/hcs2_psu.... *(1 clicks)*
pensiones *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/alojamiento/alojamientotr... *(1 clicks)*
arturo prat universidad *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/centros/centroseinstituto... *(1 clicks)*
instituto profesional santiago *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/centros/centroseinstituto... *(2 clicks)*
catolica temuco universidad *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/centros/centroseinstituto... *(1 clicks)*
cuestion social *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/orientacion/psu/hcs2_psu.... *(1 clicks)*
arquitectura *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/bibliotecas/bibliotecas_r... *(1 clicks)*
alojamiento san sebastian *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/alojamiento/alojamientotr... *(1 clicks)*
admision requisitos *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/orientacion/psu/calendari... *(1 clicks)*
trabajo *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/empresa/contratos.htm *(1 clicks)*
contratos *(frecuencia: 1)*	http://www.sitioA.cl/contenidos/empresa/contratos.htm *(1 clicks)*

Figura 12: Consultas clase B internas en el *sitio A*.

Consulta	Frecuencia
becas	58
psu	26
carreras	20
chile universidad	16
derecho	14
medicina	12
psu resultados	12
ensayos psu	11
publicidad	10
universidades	10
kinesiologia	9
chilenas universidades	9
ensayo psu	8
antropologia	8
gomez juan millas	8
institutos	8
educacion fisica	8
aranceles	8

Tabla 4: Consultas clase C en el *sitio A*.

Consulta	Frecuencia
diplomados	5
carreras vespertinas	5
test vocacional	4
beca gomez juan millas	4
becas españa	3
adolfo ibañez universidad	3
españa	3
duoc	3
calcular puntaje	3
tecnico veterinario	3
diseño interiores	3
hurtado padre	3
facsimiles	3
pedagogias	2
diferenciales ecuaciones	2
adolfo ibañez	2
diseño	2

Tabla 5: Consultas clase C' internas en el *sitio A*.

El *sitio A* tienen un gran volumen de contenidos y alto tráfico, lo que hace complicado su análisis utilizando herramientas de análisis estadístico de logs tradicionales. En este caso la herramienta de minería de consultas

Consulta	Frecuencia
diplomado electricidad gratis	3
msn	2
concursos	2

Tabla 6: Consultas clase D en el *sitio A*.

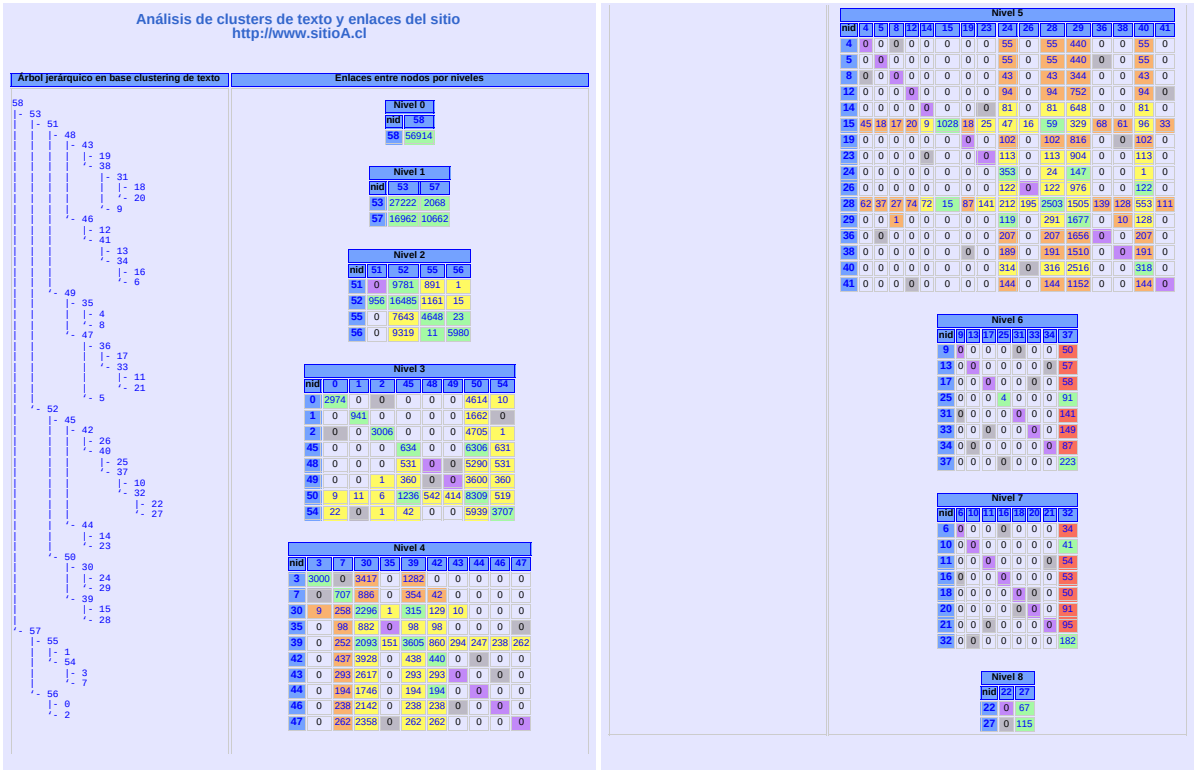


Figura 13: Análisis de clusters y enlaces en el *sitio A*.

demuestra ser de gran utilidad entregando una visión global de los intereses insatisfechos de los usuarios con respecto al sitio, además de permitir al administrador tener una noción general de la interconectividad de su sitio, lo cual no es fácil al administrar un sitio con gran cantidad de contenido. También aporta información con respecto a cuales son las consultas que posicionan a este sitio en la Web.

Para el *sitio A* en concreto, la herramienta indica conceptos y temas de interés que no están siendo abordados con suficiente profundidad al interior del sitio. Además de permitir encontrar cantidades importantes de documentos similares entre sí que no poseen conectividad entre ellos. De corregirse las situaciones anteriores, se mejora la facilidad con que los usuarios pueden encontrar temas de su interés en el sitio, lo cual disminuye el número de usuarios que abandonan en sitio insatisfechos.

6. Conclusiones y trabajo futuro

En este trabajo se presentó un modelo de análisis de consultas, estructura y contenido de un sitio, además de la implementación de una herramienta de minería de consultas. Esta herramienta permitió aplicar nuevas técnicas de análisis a un sitio Web, con el objetivo de descubrir información útil para mejorar este sitio desde el punto de vista de la percepción de los usuarios.

Este trabajo va más allá que otros modelos de minería de uso existentes hasta el momento, ya que no sólo entrega información estadística obtenida desde los logs del sitio, sino que también analiza el comportamiento de los usuarios, reflejado por la interacción que estos tienen con el motor de búsqueda interno y externo al sitio en estudio.

Los informes generados por esta herramienta indican en qué puntos el contenido del sitio puede ser mejorado, ya sea agregando nuevos temas, profundizando en los contenidos existentes o mejorando la esencia de la información utilizada al interior de éste, en particular en los textos asociados a los enlaces. A su vez, los informes hacen posible descubrir documentos de interés para los usuarios, que son difíciles de alcanzar navegando, debido a una organización interna del sitio poco intuitiva. Esta herramienta facilita la detección

de este tipo de problemas, para que así los administradores y editores puedan mejorar las falencias de sus sitios Web, haciendo más fácil y cómoda la exploración para los visitantes.

De la implementación realizada y de su aplicación en casos de uso muy diferentes, se concluye que el proceso de análisis de consultas es una forma muy útil de encontrar información novedosa sobre las necesidades y expectativas de los usuarios con respecto a un sitio Web. Adicionalmente, en los casos de uso se puede apreciar que la herramienta es útil tanto en sitios grandes como pequeños, aunque en la medida que el sitio se vuelve más grande y difícil de manejar la herramienta es más útil para tener una visión global de los puntos problemáticos en su interior. El uso de esta herramienta en régimen permanente requiere de poco esfuerzo ya que se almacena el historial de las clasificaciones previamente realizadas.

En este modelo aún existen tareas futuras por abordar. Entre estas tareas se encuentra el incorporar el trabajo de consultas similares realizado en [5], el cual permite conocer consultas similares a las consultas externas encontradas en un sitio Web. A través de las consultas similares es posible incorporar nueva esencia de la información en el sitio, para así lograr captar mayor cantidad de tráfico desde buscadores externos.

Otra forma de mejorar este modelo es incorporando nuevas funcionalidades en los procesos de análisis de la estructura y el contenido. Ésto se puede realizar profundizando el estudio de los clusters más visitados, posiblemente haciendo subclustering de estos conjuntos. También es posible realizar lematización sobre el texto del sitio, previo al clustering de los documentos, en búsqueda de mejores resultados. Adicionalmente se puede implementar la elección automática del número de clusters a utilizar en el sitio, en base al número de clusters que entregue un mejor promedio de similaridad interna y externa. La elección automática del tiempo máximo de duración de una sesión puede ser incorporada como otra posible mejora al modelo, eligiendo para cada sitio Web un tiempo adecuado a la realidad de éste.

La interfaz de la herramienta puede ser mejorada, para hacerla algo más comprensible para administradores de sitios Web. Posiblemente implementaciones futuras podrían incorporar plug-ins que permitan el funcionamiento con otros buscadores internos existentes en el mercado. Además, sería útil integrar un método para comparar diferentes resultados generados por el modelo, para así validar los cambios realizados utilizando la información entregada por la herramienta.

Referencias

[1] Akwan Information Technologies. Myweb search. `http://www.akwan.com.br`, 2005.

[2] Ricardo Baeza-Yates. Excavando la web. *El profesional de la información*, 13(1):4–10, Jan-Feb 2004.

[3] Ricardo Baeza-Yates. Web usage mining in search engines. In *Web Mining: Applications and Techniques, Anthony Scime, editor.* Idea Group, 2004.

[4] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *Web Clustering Workshop at EDBT 2004*, Crete, Greece, 2004. LNCS Springer.

[5] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines, 2004.

[6] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Ranking boosting based in query clustering. In *Atlantic Web Intelligence Conference*, Cancun, Mexico, 2004. LNCS Springer.

[7] Paulo Batista and Mario J. Silva. Mining on-line newspaper web access logs, 2002.

[8] Bettina Berendt and Myra Spiliopoulou. Analysis of navigation behaviour in web sites integrating multiple information systems. In *VLDB Journal, Vol. 9, No. 1 (special issue on "Databases and the Web")*, pages 56–75, 2000.

[9] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002.

[10] R. Cooley, P. Tan, and J. Srivastava. Websift: the web site information filter system. In *KDD Workshop on Web Mining, San Diego, CA. Springer-Verlag, in press*, 1999.

[11] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[12] Robert Cooley, Pang-Ning Tan, and Jaideep Srivastava. Discovery of interesting usage patterns from web data. In *WEBKDD*, pages 163–182, 1999.

[13] Brian D. Davison, David G. Deschenes, and David B. Lewanda. Finding relevant website queries. In *Poster Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungary, May 2003.

[14] Z. Huang, J. Ng, D. Cheung, M. Ng, and W. Ching. A cube model for web access sessions and cluster analysis. In *Proc. of WEBKDD 2001 (San Francisco CA, August 2001), 47-57.*, 2001.

[15] In-Ho Kang and GilChang Kim. Query type classification for web document retrieval. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 64–71, New York, NY, USA, 2003. ACM Press.

[16] George Karypis. CLUTO a clustering toolkit. Technical Report 02-017, Dept. of Computer Science, University of Minnesota, 2002. Available at http://www.cs.umn.edu/~cluto.

[17] F. Masseglia, P. Poncelet, and M. Teisseire. Using data mining techniques on web access logs to dynamically improve hypertext structure. *ACM SigWeb Letters vol. 8, num. 3*, pages 1–19, 1999.

[18] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. *Commun. ACM*, 43(8):142–151, 2000.

[19] O. Nasraoui and R. Krishnapuram. An evolutionary approach to mining robust multi-resolution web profiles and context sensitive url associations. *Intl' Journal of Computational Intelligence and Applications, Vol. 2, No. 3*, pages 339–348, 2002.

[20] O. Nasraoui and C. Petenes. Combining web usage mining and fuzzy inference for website personalization. In *Proceedings of the WebKDD workshop*, pages 37–46, 2003.

[21] Jian Pei, Jiawei Han, Behzad Mortazavi-asl, and Hua Zhu. Mining access patterns efficiently from web logs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 396–407, 2000.

[22] Mike Perkowitz and Oren Etzioni. Adaptive web sites: an AI challenge. In *IJCAI (1)*, pages 16–23, 1997.

[23] Mike Perkowitz and Oren Etzioni. Adaptive web sites: automatically synthesizing web pages. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 727–732, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[24] Peter Pirolli. Computational models of information scent-following in a very large browsable text collection. In *CHI*, pages 3–10, 1997.

[25] Barbara Poblete. A web mining model and tool centered in queries. M.sc. in Computer Science, CS Dept., Univ. of Chile, 2004.

[26] Masakazu Seno and George Karypis. Lpminer: An algorithm for finding frequent itemsets using length-decreasing support constraint. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 505–512. IEEE Computer Society, 2001.

[27] Ahu Sieg, Bamshad Mobasher, Steve Lytinen, and Robin Burke. Using concept hierarchies to enhance user queries in web-based information retrieval. In *IASTED International Conference on Artificial Intelligence and Applications*, 2004.

[28] Myra Spiliopoulou. Web usage mining for web site evaluation. *Commun. ACM*, 43(8):127–134, 2000.

[29] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.

[30] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Wei-Ying Ma, and Chao-Jun Lu. Log mining to improve the performance of site search. In *WISEW '02: Proceedings of the Third International Conference on Web Information Systems Engineering (Workshops) - (WISEw'02)*, page 238, Washington, DC, USA, 2002. IEEE Computer Society.

Protocolo de Coherencia de Memoria Cache para Sistemas Distribuidos

Jose L. Aguilar Castro
Universidad de Los Andes, Escuela de Ingeniería de Sistemas, CEMISID
Mérida, Venezuela, 5101
aguilar@ula.ve

y

Rodolfo L. Sumoza Matos
Universidad Pedagógica Experimental Libertador, Dirección de Informática.
Caracas, Venezuela, 1030
rsumoza@upel.edu.ve

Resumen

Este trabajo propone un protocolo que gestiona la coherencia a nivel de la memoria cache en sistemas con memoria distribuida. La memoria cache se encuentra distribuida a través de los diferentes nodos del sistema, por lo que se debe mantener coherente la información almacenada en ellas. El protocolo se validó utilizando una metodología basada en la descripción formal utilizando una máquina de estados finitos y la herramienta de software Spin.
Palabras claves: Memoria Cache, Protocolos de Coherencia, Sistemas Distribuidos

Abstract

This work proposes a protocol which manages the coherence in the cache memory in systems with distributed memory. Cache memory is distributed in different system's nodes, for this reason stored information in them, must to be maintained coherent. This protocol was proved using a methodology based in the formal description using a finites states machine and the software Spin tool.
Key words: Cache Memory, Coherence Protocols, Distributed Systems

1. Introducción

El enfoque de este trabajo se basa en una plataforma de memoria distribuida, tanto a nivel de la memoria cache, como de la memoria principal. Principalmente se han realizado estudios en cuanto a la coherencia de la cache cuando se utiliza memoria compartida [1, 2, 3, 6, 8, 9, 10, 13]. En cuanto a memorias distribuidas, hasta ahora la mayoría de los trabajos se han basado en plantear la distribución de la memoria principal, y su interacción con sus respectivas cache, y no abordan el tema de la cache distribuida específicamente [2, 7, 8]. Se nombraran a continuación algunos de los trabajos que tratan el problema de coherencia, pero para sistemas a memoria compartida distribuida. Uno de los modelos que ha sido aplicado para resolver el problema de coherencia en sistemas a memoria distribuida compartida es el esquema basado en *directorio*. Un ejemplo de implementación de este tipo de ideas es la máquina denominada *DASH* [10], la cual utiliza directorios distribuidos. Otra propuesta basada en directorios que reduce la latencia remota de datos, es la denominado COMA (Cache Only Memory Arquitecture) [7]. En [11, 12] Xiaowei Shen y Larry Rudolph, presentan un protocolo adaptativo que intenta superar los inconvenientes generados en el mantenimiento de la coherencia de datos, llamado *Cachet*. El trabajo de J. Archibald y J. Baer propone un esquema de coherencia, el cual implementa un directorio centralizado (global) que utiliza sólo dos *bits* para registrar los estados de las direcciones de memoria copiadas en la cache [5]. Anant Agarwal y richard Simoni, evaluaron los dos esquemas principales para la solución de la coherencia en cache: El esquema basado en Snoopy y el basado en .directorio. Thomas Alexander y Gershon Kedem en [4] propusieron un diseño de cache y buffer con prebúsqueda distribuida parea sistemas de memoria de alto rendimiento. John Hennessy, Mark Heinrich y Anoop Gupta en [10] describieron las perspectivas que se desarrollan alrededor del problema del mantenimiento de la coherencia en cache para los Sistemas con Memoria Compartida y Distribuida.

Este trabajo propone un protocolo que gestiona la coherencia a nivel de la memoria cache en sistemas con memoria distribuida, en el cual también la memoria cache esta distribuida. Este trabajo se organiza como sigue, a continuación

presentamos el marco teórico usado, después presentamos el protocolo propuesto. Finalmente se presentan las verificaciones hechas y las conclusiones.

2. Marco Teórico

2.1. Memoria Cache

Es una memoria temporal, generalmente de existencia oculta y automática para el usuario, que proporciona acceso rápido a los datos de uso más frecuente. El objetivo de utilizar la memoria cache es aumentar el rendimiento de la memoria principal, duplicando áreas de datos en módulos de rápido acceso. La forma de trabajar de la memoria cache es simple, los datos pedidos por el procesador son buscados primero en la memoria cache, y después, si no están disponibles en ella, se buscan en la memoria principal.

Ahora bien, los entornos donde la memoria está distribuida entre los procesadores, implican que cada procesador tiene acceso a su propia memoria. Esto conlleva a que la gestión y administración de los datos es más compleja ya que cuando los datos a usar por un procesador están en el espacio de direcciones de otro, será necesario solicitarlos y transferirlos a través de mensajes.

Existen una serie de mecanismos para la administración y gestión de la memoria cache que son utilizados por los distintos tipos de plataformas distribuidas (memoria compartida, memoria distribuida, etc.), los cuales influyen directamente en el rendimiento y el costo de este recurso. Estos mecanismos son [2, 4, 6, 8]:

- **Mecanismos predictivos.** Este esquema se basa en almacenar en la cache datos que probablemente se utilizaran a posterior. Se intenta predecir su uso futuro.
- **Mecanismo Write Through.** Es un mecanismo en el cual la data es escrita en la memoria principal al mismo tiempo que es almacenada en la memoria cache.
- **Mecanismo Write Back (Wb).** La nueva información sólo es escrita en la memoria cache, y cuando el espacio en memoria cache es necesitado por alguna otra dirección de memoria, la data cambiada es escrita en la memoria principal.

La distribución de la memoria cache en sistemas con memoria principal distribuida, significa que cada procesador posee su propia cache y su propia memoria principal. Los mecanismos presentados previamente establecen una relación entre la memoria principal y su cache, sin considerar a los otros nodos del sistema. Pero que sucede cuando los datos están replicados a través de las diferentes memorias caches?. Allí el problema de coherencia deviene crucial.

2.2. Coherencia

La coherencia significa que cualquier lectura debe retornar el valor de la escritura más reciente. Mientras más estricta sea esta idea, más difícil de implementar será. Para explicarlo de otra forma, la coherencia se logra cuando cualquier escritura debe ser vista por una lectura, y todas las escrituras son vistas en el orden apropiado (serialización). Existen dos reglas para lograr lo anterior:

a) Si un procesador P escribe el valor x en una localidad de memoria, y luego un procesador P1 lo lee, las escrituras de P serán vistas por P1.
b) Las escrituras a una misma localidad de memoria están serializadas. Así, la última escritura es la única que debe ser siempre vista por cualquier sitio que la solicite.

Para el mantenimiento coherente de las cache se utilizan básicamente dos tipos de protocolos, que ha continuación se mencionan:

a) Protocolo Entrometido, de Sondeo o Snoopy: Su mecanismo básico es el de distribuir la información permanentemente usando las políticas de la forma siguiente [2, 8]:

- Invalidación en Escritura o Escribir-Invalidar: se escribe el nuevo dato donde se genera la solicitud y se invalidad sus copias.
- Actualización en Escritura, Escribir-Actualizar: se escribe el nuevo dato donde se genera la solicitud y se actualizan sus copias.

b) Protocolo con Directorio (mantiene la coherencia utilizando la memoria principal): Un Protocolo basado en Directorios implementa una estructura, llamada tabla o directorio, que registra cuál procesador ha gravado en su

memoria cache cualquier bloque dado de la memoria principal [5]. Este tipo de protocolos utiliza el directorio para mantener una visión consistente de los datos [2]. También se han utilizado protocolos que no implementan un directorio global o centralizado, sino que permiten que cada nodo posea su propio directorio [2].

3. Presentación de la Propuesta

3.1 Generalidades

Se propone el diseño de un protocolo para la gestión de la coherencia en sistemas distribuidos, con memoria principal y memoria cache distribuidas. Del entorno del que se está hablando es uno que utiliza la conexión en red de un conjunto de nodos, donde cada uno representa un computador con un CPU, una memoria cache y una memoria principal, conectado a través de una red de interconexión con el resto de los nodos.

Cada nodo, inicialmente tiene un conjunto de datos en su memoria principal, y para ese conjunto de datos, ese nodo representa su nodo hogar o Home. Además cuenta con un directorio local que registra ese conjunto de datos que pueden estar almacenados en memoria cache local o remota. El nodo hogar es el dueño de ese conjunto de datos, por ende, ningún otro nodo puede tener algún dato de ese conjunto en su memoria principal, pero si en su memoria cache.

El protocolo propuesto utiliza como base para la gestión de la consistencia el paradigma de Directorios Distribuidos (ver figura 1), implementados localmente en cada nodo a través de Árboles Binarios Balanceados, como estructura dinámica (ver figura 2), y como mecanismo de coherencia se utiliza la Actualización en Escritura (Write update), y un mecanismo de exclusión mutua distribuida, la cual está implícita en el procedimiento. En la Actualización en Escritura que nosotros implementamos, cuando un procesador cambia un dato este mecanismo actualiza de forma inmediata la memoria principal y la memoria cache local donde pertenece el dato. En el caso de actualizaciones remotas, la cache remota queda actualizada junto con la cache y la memoria principal del nodo al que pertenece el dato (nodo hogar). En todos los casos, el resto de las memorias cache de los otros nodos que comparten el dato y que están registrados en el directorio que se encuentra en el nodo hogar, son invalidadas.

Cada dato posee tres (3) estados posibles: Limpio, Inválido y Bloqueado. El primer caso representa un dato que no ha sido modificado por algún procesador desde su última actualización. El segundo caso ocurre cuando un dato ha sido modificado por algún otro nodo, tal que a su vez se actualiza la memoria en el nodo hogar y se invalida el resto de los nodos que comparten el dato. El tercer caso ocurre cuando un nodo intenta modificar un dato, y esta acción no se ha completado en la memoria principal del nodo hogar; así, el estado en la memoria cache del nodo el cual realiza la actualización será el de bloqueado; de esta forma, ningún otro proceso puede acceder a este dato hasta terminar la acción que se está realizando sobre él.

Cada directorio tiene los siguientes campos:

- Campo de Referencia del Dato o crd: Es un apuntador al dato que se almacena en la memoria, y es el índice de búsqueda del directorio. Para los casos en los cuales los datos no pertenecen al nodo**,** dicho apuntador será la dirección IP del nodo hogar del dato**.**
- Campo de Información del Estado o cie: Contiene la información del estado actual del dato (Limpio, Inválido o Bloqueado).
- Campo del identificador Remoto del Dato o ivr: Es el nombre que le asigna el nodo remoto al dato. Este se utiliza con la finalidad de evitar inconsistencias a nivel de nombres de datos locales y remotos. Para el caso cuando el dato pertenece a ese nodo el valor de este campo es el mismo que el campo ivl.
- Campo del Identificador Local del Dato o ivl: Es el nombre que le asigna el nodo local al dato. También se utiliza con la finalidad de evitar inconsistencias a nivel de nombres de datos remotos y locales.
- Campo de la dirección de memoria cache o dcl: Identifica el sitio en la memoria cache donde se encuentra el dato.
- Campos de información de la presencia del Dato o cid1 y cid2: Son dos campos utilizados para construir el Árbol Binario, que contienen la información (las direcciones IP) del resto de los nodos (computadoras) que en un momento dado comparten el dato (ver figura 2). Cada uno de los "nodos" del Árbol Binario posee tres campos: Uno indica la dirección IP del nodo, el otro es un conmutador utilizado para el balanceo, y el tercero es un par de apuntadores a sus "nodos" hijos.
- Campo conmutador conm: Refleja la última asignación hecha en el árbol, es decir, si fue en su rama izquierda o derecha. Se utiliza para fines del balanceo de dicho árbol.
- Cuando el dato es remoto, es decir no pertenece al nodo, los campos conm, cid1 y cid2 son nulos.

El esquema de dicho protocolo se presenta como la interacción entre el procesador, la memoria principal, y la memoria cache de los nodos que conforman el sistema. La comunicación entre nodos se realiza a través de pases de mensajes administrados segun una política FIFO.

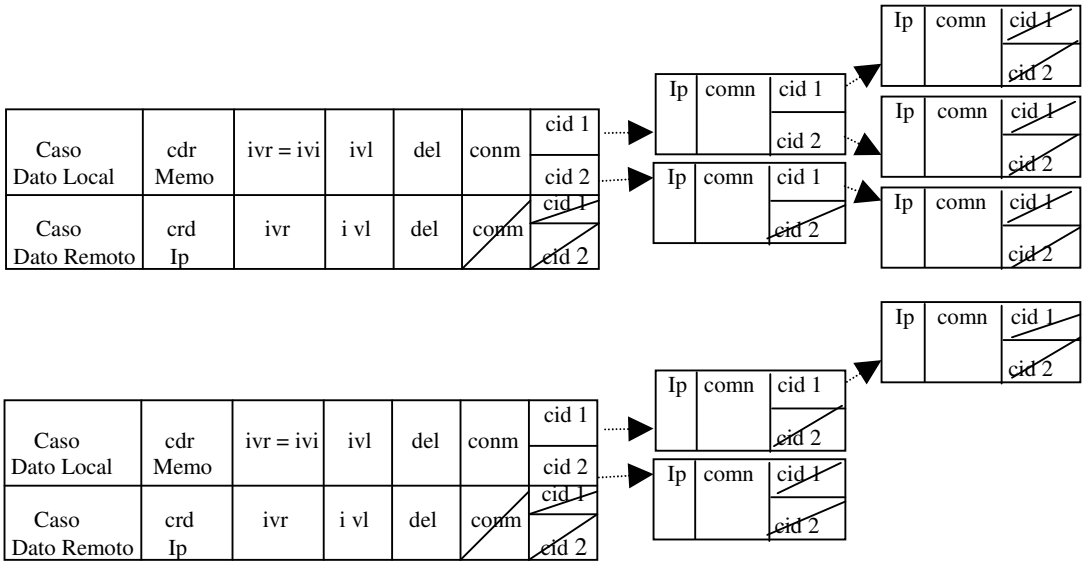


Figura 1. Estructura del Directorio

3.2. Macroalgoritmo

A. Al momento que se genera una solicitud de un dato por parte de un procesador:
   i. En primera instancia el dato se busca en la cache local, si se encuentra y está en el estado Limpio, se le suministra al procesador.
   ii. Si el dato se encuentra en la memoria cache local y posee el estado Inválido, se genera una transacción de lectura hacia el nodo hogar (si el dato se encuentra en estado inválido quiere decir que ese dato no pertenece a ese nodo).
   iii. Cuando el dato se encuentra en el estado bloqueado, se niega la petición de lectura.
   iv. Si el dato no se encuentra en la memoria cache, es decir, no se encuentra en el directorio, entonces se busca en la memoria principal local, si se encuentra, entonces se almacena en memoria cache y se actualiza el directorio.
   v. Si el dato no se encuentra en la memoria principal, se genera una transacción de lectura sobre la red de interconexión.

B. Realizar uno de los siguientes pasos, por parte del nodo hogar, cuando se genera una transacción de lectura sobre la red de interconexión:
   i. Si el dato está en su memoria cache, y en el estado Limpio, se coloca en la red de interconexión, para dárselo al nodo solicitante, y se actualiza el directorio (árbol) del nodo hogar, agregando la dirección IP del nodo solicitante.
   ii. Si el dato está en su memoria cache y en estado Bloqueado, se envía un mensaje invalidando la transacción de lectura solicitada.
   iii. Cuando el dato no está en la memoria cache del nodo hogar, se copia desde la memoria principal al cache local, y se envía a través de la red de interconexión al nodo solicitante, y se actualiza el directorio.
   iv. Cuando un nodo remoto almacena en su memoria cache algún dato perteneciente a otro nodo, lo registra en su directorio colocándole la dirección IP del nodo hogar.

C Tramitar de la siguiente forma los casos donde se generen peticiones de escritura por parte de un procesador:
   i. Cuando el dato se encuentra en la cache local, y pertenece a ese nodo, se realiza la escritura a través del mecanismo Write Update, el cual actualiza la memoria principal, y se invalidan las copias del dato encontradas en las memorias cache de los nodos registrados en el directorio. Este mecanismo bloquea el dato mientras se termina el proceso de actualización en la memoria principal local. Se elimina el árbol de registro de ese dato en el directorio del nodo hogar. Al terminar, se desbloquea el dato en la cache del nodo hogar.
   ii. Cuando el dato se encuentra en la cache local, y no pertenece a ese nodo, se bloquea el dato de ese nodo

remoto y se envía un mensaje al nodo Hogar, para que realice el mecanismo Write Update, el cual actualiza su memoria principal, su memoria cache, y la memoria cache del nodo que realizó la modificación queda actualizada. Además, se invalidan el resto de las copias del dato almacenadas en las memorias cache de los nodos registrados en el directorio, hecho que incluye la eliminación del árbol que registra y apunta hacia dichos nodos, exceptuando al nodo que realizó la modificación. Una vez concluido el proceso de actualización, se envía un mensaje de desbloqueo del dato en el nodo remoto.

D Manejar las solicitudes concurrentes de dos o más nodos de la siguiente forma.

i. Cuando las solicitudes de escritura se generan a la vez desde distintos nodos hacia un nodo específico, se escoge aleatoriamente uno de ellos para ser procesado, y se invalidan el resto de los nodos involucrados. En otras palabras, para los casos donde existan colisiones, es decir, si dos nodos intentan hacer una escritura simultánea, entonces se escoge aleatoriamente uno de ellos y al resto se les envía un mensaje de invalidación de la transacción.

ii. Cuando las solicitudes son de lecturas, y se realizan de forma simultánea, se genera una cola y se atienden según la política FIFO (First in, First Out).

En la figura 2 se muestra la máquina de estados finitos que describe el comportamiento del protocolo a nivel de la memoria cache, la cual representa el conjunto de estados de la memoria cache. El modelo contiene las transiciones entre los estados, derivados por los eventos generados por los componentes de los nodos del sistema (procesadores, memorias cache y memorias principales). A continuación describiremos las reglas que describen el comportamiento de cada componente de cada nodo (procesador, memoria principal y memoria cache).

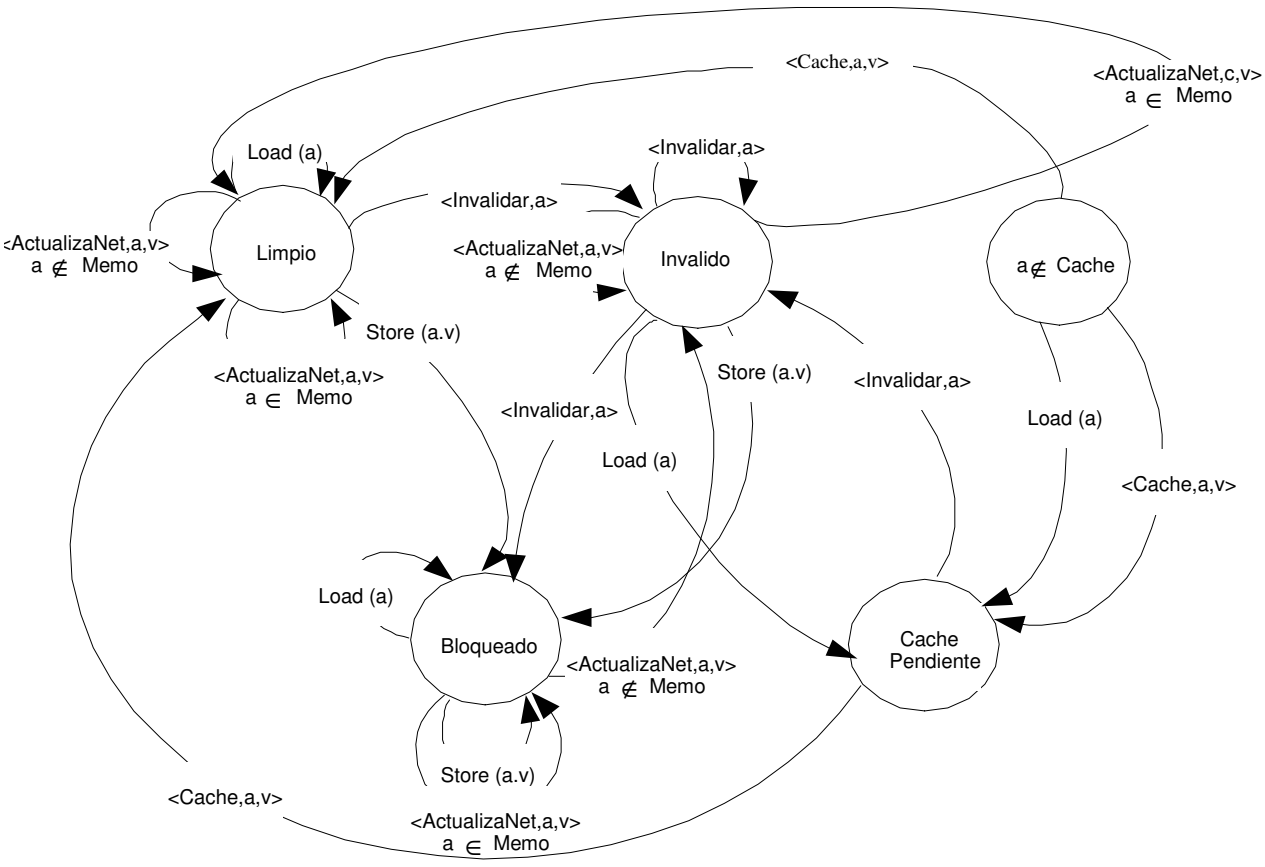


Figura 2. Máquina de Estado Finitos

3.2.1. Reglas para las solicitudes del Procesador

La siguiente tabla muestra el conjunto de reglas asociadas al procesador:

Reglas del Procesador			
Instrucción	Edo. Inicial Cache	Acción	Edo. Final Cache
Load(a)	Celda(a,v,Limpio)	Finalizar Load	Celda (a,v,Limpio)
	Celda(a,v,Invalido)	<CacheSol,a> → Memo	Celda(a,-CachePendiente)
	Celda(a,v,Bloqueado)	Finalizar Load	Celda(a,v,Bloqueado)
	a ∉ cache	<CacheSol,a> → Memo	Celda(a,-,CachePendiente)
Store(a,v)	Celda(a,-,Limpio)	<Wu,a,v> → Memo	Celda(a,v, Bloqueado)
	Celda(a,-,Invalido)	Finalizar Store	Celda(a,v, Invalido)
	Celda(a,v1,Bloqueado)	Finalizar Store	Celda(a,v1,Bloqueado)

Tabla 1. Reglas del Procesador

Explicación y notación:

- ***Instrucción (a,v)***, representa una instrucción que puede ser: *Store(a,v), Load(a), Reconcile(a),* que utilizan la dirección de memoria *a* y el valor del dato *v* (puede ser opcional, según sea el caso). Cada una significa lo siguiente:
  *Load: Solicitud de lectura de un dato.
  *Store: Solicitud de escritura de un dato.
- En cuanto a los estados de la cache, ***Celda (a,v,Estado)*** representa una Celda o Bloque en la memoria cache con dirección *a,* valor del dato *v* (opcional) y su estado actual que puede ser Limpio, Inválido o Bloqueado.
- La acción que se representa con un patrón como el siguiente: ***<cmd,a,v>***, simboliza un mensaje cuyo comando asociado es *cmd*, con dirección en memoria principal *a* y valor del dato *v* (puede ser opcional, según el caso). Para el caso de las reglas del procesador, <cmd,a,v> → Memo, quiere decir que el mensaje se ha enviado a la memoria principal (Memo). El comando es la parte que indica qué es lo que el mensaje está requiriendo. Los mensajes pueden ser ***<Wu,a,v>*** y ***<CacheSol,a>.***
- ***<Wu,a,v>*** representa un mensaje de solicitud de escritura o actualización Write Update, a la dirección de memoria *a* con el valor del dato *v*.
- ***<CacheSol,a>*** representa un mensaje que solicita el valor o dato que posee la dirección de memoria *a*.
- Un estado ***CachePendiente*** es un estado momentáneo que representa la espera de la respuesta a la solicitud hecha a la memoria de un dato.
- a ∉ cache , quiere decir que la dirección de memoria *a* no se encuentra almacenado en la Cache.
- ***Reconcile(a)*** elimina físicamente los datos de la memoria cache local. Esta instrucción sólo se utiliza para los casos en que la memoria cache esté llena.

Una instrucción *Load* finaliza después que el dato ha sido enviado al procesador, y una instrucción *Store* se finaliza después que la memoria principal y la cache han sido modificadas en el nodo hogar.

3.2.2. Reglas de la Cache y de la Memoria Principal

Las reglas que establecen el funcionamiento lógico del protocolo en la memoria cache se muestra en la tabla siguiente:

Reglas Mandatorias del Motor de Cache			
Mensaje de (*)	Edo. inicial cache	Acción	Próximo edo. cache
<Cache,a,v>	a ∉ cache		Celda(a,v,Limpio)
	Celda(a,-,CachePendiente)		Celda(a,v,Limpio)
<Invalidar,a>	Celda(a,-,Limpio)		Celda(a,-,Invalido)
	Celda(a,v,Inválido)		Celda(a,-, Invalido)
	Celda(a,-,Bloqueado)		Celda(a,-, Invalido)
	Celda(a,-,CachePendiente)		Celda(a,-, Inválido)
<ActualizaNet,a,v>	Celda(a,-,Limpio) a ∈ Memo		Celda(a,-,Limpio)
	Celda(a,v1,Inválido) a ∈ Memo		Celda(a,-,Limpio)
	Celda(a,v,Bloqueado) a ∈ Memo		Celda(a,v,Bloqueado)
	Celda(a,-,Limpio) a ∉ Memo	<Invalidar,a>→dir	Celda(a,-, Inválido)
	Celda(a,v1,Inválido) a ∉ Memo	<Invalidar,a>→dir	Celda(a,-, Inválido)
	Celda(a,v,Bloqueado) a ∉ Memo	<Invalidar,a>→dir	Celda(a,-, Inválido)

(*)Memoria Principal del nodo hogar

Tabla 2. Reglas de la Cache

Las reglas de la memoria principal son las siguientes:

Reglas Mandatorias del Motor de Memoria			
Mensaje	Edo. inicial de memoria	Acción	Próximo edo. memoria
<CacheSol,a>	Celda(a,-,D[dir]) a ∉ dir	<MemoSol,a>→Memo	Celda (a,v,MemoPendiente)
	Celda(a,-,D[dir]) a ∈ dir	<Cache,a,v>→id	Celda (a,v,D[dir])
<Wu,a,v1>	Celda(a,v, D[dir]) a ∈ Memo	<Invalidar,a>→dir-id	Celda(a,v1, D[id])
	Celda(a,v, D[dir]) a ∉ Memo	<ActualizaNet,a,v>→id	Celda(a,v, D[id|dir])
<MemoSol,a,>	Celda(a,-,D[dir]) a ∈ Memo	<Cache,a,v>→id	Celda(a,v,D[id|dir])
	Celda(a,-,D[dir]) a ∉ Memo	<MemoNet,a,v>→Net	Celda(a,v,NetPendiente)
<MemoNet,a,v>	Celda(a,-,Memo) a ∈ Memo	<Cache,a,v>→id	Celda(a,v,D[id|dir])
	Celda(a,-,Memo) a ∉ Memo	Finaliza	a ∉ Memo

Tabla 3. Reglas de la Memoria

Explicación y Notación:

- El origen y el destino de un mensaje pueden ser un sitio en cache ó *id* (es decir representa su dirección IP), un conjunto de sitios cache o nodos (*dir*), la memoria principal local o remota (*Memo*), o el procesador.
- Un mensaje recibido en la memoria principal es siempre enviado del sitio *id,* del procesador*,* de una memoria remota o de la misma memoria local, mientras que un mensaje recibido en la cache es siempre enviado desde la memoria principal local o remota.
- ***<cmd,a,v>-->dir*** quiere decir el envío del mensaje con un comando cmd a un conjunto de sitios de cache (dir), con dirección de memoria a y valor del dato v (opcional).
- **<cmd,a,v>-->Memo,** a ∉ Memo **, <Wu,a,v>, CacheSol, CachePendiente** tienen la misma significación que en las reglas del procesador.
- **<Cache,a,v>** representa un mensaje que le indica a la cache que almacene el valor o dato v con dirección de memoria a.
- **<Invalidar,a>** representa un mensaje que le solicita a la cache que invalide su contenido o dato, y elimina el apuntador en el directorio del nodo hogar a ese dato, de tal forma que el espacio del dato inválido pueda ser reutilizado.
- Para el caso de la memoria **Celda(a,v, D[dir])**, representa una celda en la memoria principal con dirección de memoria a, valor del dato v, y con un directorio el cual contiene almacenado el conjunto de referencias a nodos dir.
- **Celda(a,v,D[id|dir]),** también relacionado a la memoria, representa una celda en la memoria principal con dirección a, valor del dato v, y con un directorio el cual contiene el conjunto de referencias a nodos dir, en el que se incluye la referencia id, de forma específica.
- **Celda (a,v,MemoPendiente)** Representa una celda en la parte de la memoria principal que almacena el directorio, la cual posee un estado de espera por una respuesta que debe provenir de la otra parte de la memoria, pero fuera del directorio, es decir, en su contenido general.
- **Celda(a,v,NetPendiente)** Representa una celda en la parte de la memoria principal que almacena el directorio, la cual posee un estado de espera por una respuesta que debe provenir de la red (de otro nodo).
- **<MemoSol,a>** Representa un mensaje que envía la parte de la memoria que posee el directorio, a la parte de la memoria general de la memoria.
- **<MemoNet,a,v>** Representa un mensaje enviado por la memoria principal a la red, solicitando un dato con dirección a y valor v.
- **<ActualizaNet,a,v>** Es un mensaje enviado por la red, el cual solicita la actualización del dato en los nodos. Para este caso en particular, si es el nodo hogar actualiza el dato, de lo contrario lo invalida.
- Cada regla de la memoria, cuando está asociada a un conjunto de referencias a nodos (con sus memorias cache), es decir, cuando envía un mensaje a un dir, la memoria lo distribuye a todos los nodos, y cada nodo lo gestiona de forma local.

4. Validación

La validación se realizó de dos formas, la primera, a la que se le dio el nombre de validación analítica, consiste en verificar analíticamente que en todos los procesos generados por el protocolo se obtenga siempre alguna respuesta (criterio de completitud). La segunda forma de validación consiste en evaluar algunos criterios de correctitud en forma práctica, es decir, criterios que permitan establecer si el protocolo realiza todos los procesos de forma correcta.

4.1. Validación Analítica

En esta parte se evaluará el criterio de "completitud". La completitud consiste en una propiedad de los sistemas relacionada a la capacidad de finalizar de forma adecuada cada uno de los procesos que inicia, es decir, que cada proceso iniciado debe ser finalizado, y de una forma que se considere correcta. Para esto, en el primer tipo de evaluación para verificar la consistencia lógica se utilizó la máquina de estados finitos de la figura 2. Este estudio se basa en establecer que cada uno de los estados que se representan pueden ser alcanzados de alguna manera por la dinámica del sistema, ya que en caso contrario existirían estados no alcanzables que determinarían un error de diseño, lo que implicaría que parte de la estructura de protocolo nunca se ejecutaría. Así, la evaluación de completitud consiste en establecer de forma gráfica cada uno de los caminos posibles que pueden tener los procesos que se pueden desarrollar utilizando el protocolo propuesto en este trabajo.

En relación a los estados de la memoria cache, y para el modelo representado por la máquina de estados finitos (figura 3), se tienen los estados de transición siguientes (estados que acontecen en un momento muy corto o sólo refleja resultado de una consulta a la memoria cache):

- a ∉ cache.
- CachePendiente.

Estos estados de transición se consideran en el análisis debido a que a través de ellos se logra completar la dinámica del protocolo, es decir, por medio de los estados de transición se puede llegar de un estado real a otro. Los estados reales son los siguientes:

- Limpio.
- Inválido.
- Bloqueado**.**

Este modelo representa la máquina de estados finitos de un sólo nodo, y la dinámica se replica de igual manera para el resto de los nodos. El modelo del protocolo se sometió a un análisis exhaustivo de los estados que pueden o no ser alcanzados, a través de un algoritmo exhaustivo propuesto en [9]. Esta verificación es simple, aunque exhaustiva, ya que consiste en recorrer la máquina de estados finitos comenzando por un estado cualquiera y se van chequeando el resto de los estados que pueden ser alcanzados. Por ejemplo, si se escoge de forma aleatoria como estado inicial al estado *Limpio*, en la quinta iteración se logra alcanzar todos los estados del sistema, por ende no contiene ningún error básico de diseño según esta forma de evaluar el criterio de completitud.

La segunda forma de evaluación de la completitud consiste en evaluar de forma gráfica que cada instrucción generada por un procesador (load, store) se ejecute en su totalidad. Esta evaluación se basa en el seguimiento de las reglas y dinámicas especificadas en las tablas 1, 2 y 3, las cuales muestran las reglas del procesador, de la memoria principal y cache.

La condición para evaluar la completitud se considera en base al seguimiento de cada solicitud hecha por el procesador, chequeando que el protocolo tenga para cada instrucción una ejecución completa. Tal como se muestra en las tablas 1, 2 y 3, al realizar un seguimiento de cada uno de los procesos ejecutados por un procesador, la culminación de cada ejecución o proceso debe ser llevado a cabo por una acción que indique la finalización de la instrucción de forma directa (por ejemplo, "F*inalizar Load*") o eliminando cualquier estado de transición (por ejemplo, CachePendiente), a través del desarrollo de cada uno de los pasos que se señalan en dichas tablas.

La figura 3, es un diagrama de flujo utilizado para la verificación de completitud, que representa la ejecución de la instrucción Load por parte del procesador cuando el estado de la cache es Inválido. Cada óvalo representa un estado de la cache, cada rombo representa una situación de decisión donde, dependiendo de algún valor se determina el siguiente paso; y el texto entre líneas representa algún mensaje o una instrucción del procesador.

En esta figura, el primer óvalo de la izquierda representa el inicio del recorrido con un estado de cache Inválido. De este óvalo se llega al siguiente, el cual representa un estado CachePendiente, a través de la instrucción Load (mostrada entre líneas). Al generarse el mensaje <cachesol, a> (solicitud de la dirección "a" a la cache), se llega a un estado de decisión en el cual se determina si la dirección está o no actualmente almacenada en el directorio. Si la respuesta es afirmativa, se envía un mensaje al motor de la cache, para solicitarle la ejecución de una tarea, la cual depende del estado actual de la cache (CachePendiente o "a" ∉ cache). Para cualquiera de los casos, el nuevo estado de la cache es el estado Limpio.

Si la dirección "a" no hubiese estado en el directorio, se genera una solicitud a memoria a través del mensaje <MemoSol, a>, para lo cual se debe saber si la dirección "a" existe o no en la memoria principal. Si es afirmativa la respuesta, se envía un mensaje <cache,a,v> al motor de la cache para pasar al estado Limpio, ya sea que esté en el estado CachePendiente o si "a" ∉ cache. Si la dirección "a" ∉ Memoria principal, entonces se envía el mensaje <MemoNet,a,v> a través de la red, solicitando al resto de los nodos la dirección requerida.

Si "a" no se encuentra en las memorias remotas, el proceso se finaliza, en caso contrario se envía un mensaje al motor de la caché para que pueda colocarse en el estado Limpio, sustituyendo el estado Cachependiente o "a" ∉ cache.

Lo anterior demuestra que el protocolo genera una culminación adecuada para cada caso considerado, por ende, es completo.

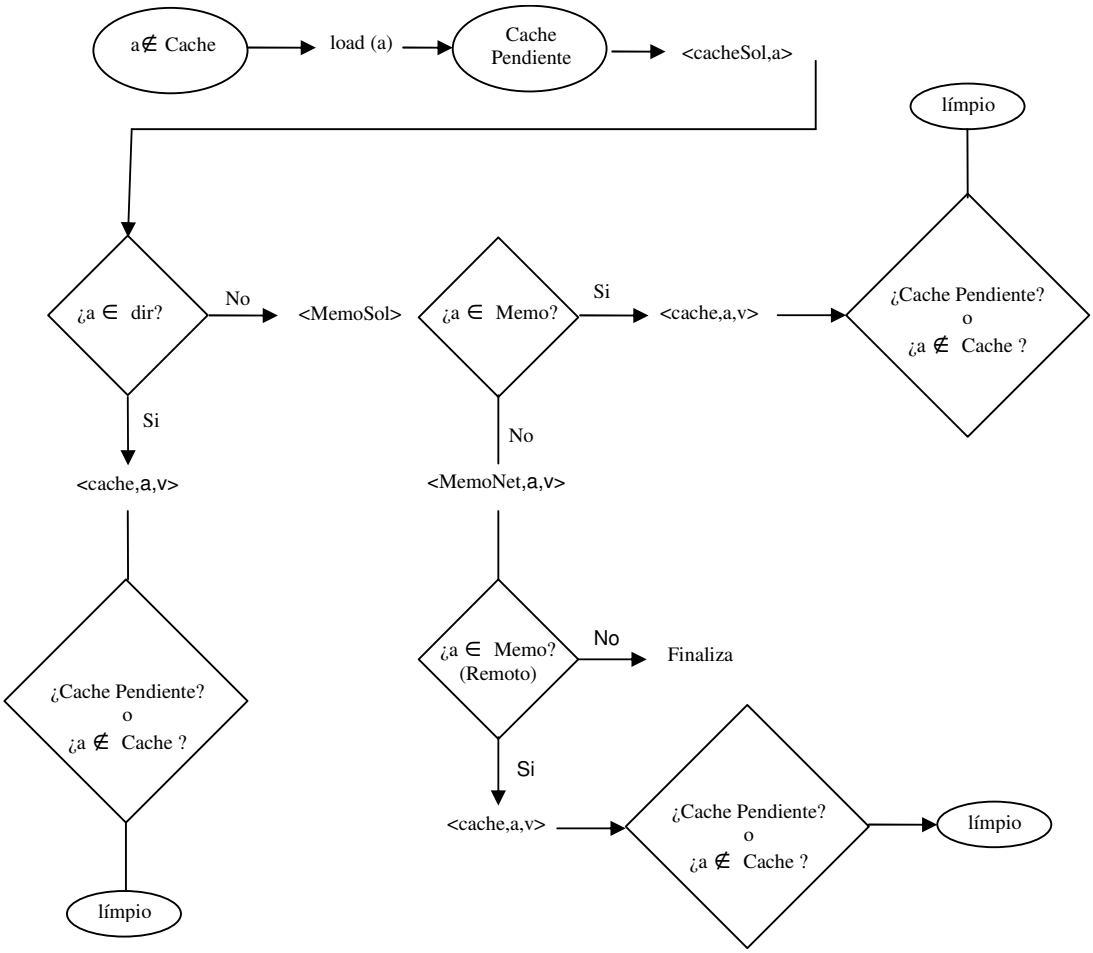


Figura 3. Instrucción Load, con estado de la cache: a ∉ cache

4.2. Validación Experimental

En esta parte se evaluarán criterios de "correctitud". Consiste en encontrar de forma experimental un error en el diseño que impida la ejecución esperada y adecuada del protocolo. La herramienta utilizada se denomina Spin, la cual es un interpretador de un Meta Lenguaje denominado PROMELA (Process Meta Languaje) [14].

Para el modelo de esta investigación, sólo se consideraron criterios de correctitud relacionados con el diseño de protocolos de comunicación de datos en Sistemas Distribuidos. Estos criterios son: Verificación de la no Existencia de abrazos mortales y/o lazos infinitos. Inicialmente se elaboró un modelo del protocolo propuesto utilizando PROMELA, en el cual se consideraron sólo dos nodos, cada uno con su procesador, su memoria cache y su memoria principal. Un ejemplo de los resultados obtenidos de las simulaciones ejecutadas fue:

#processes: 5

queue 1 (deCPU1):
queue 3 (alCPU1):
queue 6 (aRed1): [solicitud1,2,0]
queue 2 (deCPU2):
queue 4 (alCPU2):
queue 5 (aRed2): [solicitud2,1,0]
L = 1
I = 2

```
                B = 3
                CP = 4
                NEC = 5
 37:    proc  4 (Memo2) line 155 "ACacheSpin" (state 58)
 37:    proc  3 (Memo1) line  78 "ACacheSpin" (state 58)
 37:    proc  2 (cpu2) line  47 "ACacheSpin" (state 4)
 37:    proc  1 (cpu1) line  35 "ACacheSpin" (state 6)
 37:    proc  0 (:init:) line 210 "ACacheSpin" (state 6) <valid end state>
5 processes created
```

La explicación de la ejecución anterior es la siguiente:
#processes: 5 Esta línea indica que se ejecutaron 5 procesos.

```
        queue 1 (deCPU1):
        queue 3 (alCPU1):
        queue 6 (aRed1): [solicitud1,2,0]
        queue 2 (deCPU2):
        queue 4 (alCPU2):
        queue 5 (aRed2): [solicitud2,1,0]
```

Estas líneas representan las colas que se forman en cada uno de los canales utilizados para la transmisión de mensajes. Es decir, la política utilizada en PROMELA para la transmisión de mensajes, es la de crear canales que contengan estructuras internas capaces de albergar las colas de los mensajes entrantes y salientes. Para este caso deCPU1, alCPU1, aRed1, deCPU2, alCPU2 y aRed2 son canales a los cuales se le asignaron los números de colas 1,3,6,2,4,5, respectivamente. Para el caso de la cola 6 y 5, se indica que al final de la simulación quedaron en esas colas los mensajes [solicitud1,2,0] y [solicitud1,1,0], respectivamente. *L, I, B, CP y NEC* son variables auxiliares. Las líneas:

```
 37:    proc  4 (Memo2) line 155 "ACacheSpin" (state 58)
 37:    proc  3 (Memo1) line  78 "ACacheSpin" (state 58)
 37:    proc  2 (cpu2) line  47 "ACacheSpin" (state 4)
 37:    proc  1 (cpu1) line  35 "ACacheSpin" (state 6)
 37:    proc  0 (:init:) line 210 "ACacheSpin" (state 6) <valid end state>
```

hacen referencia a los procesos ejecutados, numerados del 0 al 4. Además, al proceso "*init*" se le indica que tuvo un estado de finalización válido, es decir, se logró llegar al fin de la ejecución de forma válida. Existe otra información que en nuestro caso no es de interés.

El análisis de lo anterior muestra que el modelo ejecutó en la simulación 5 procesos. Pero lo más importante es que al final se observa un indicador que menciona que el modelo posee un estado válido de finalización (*valid end state*). Lo que quiere decir es que al ejecutar los procesos aplicando cada una de las reglas del protocolo no se generaron abrazos mortales, ni lazos infinitos, ya que de lo contrario la simulación no hubiese finalizado.

Por otro lado, se utilizó Spin como un verificador exhaustivo, capaz de verificar rigurosamente la validez de los requerimientos de correctitud especificados. Dicho proceso de verificación se muestra a continuación:

```
Full statespace search for:
        never claim         - (none specified)
        assertion violations        -
                acceptance   cycles             + (selected)
                invalid end states   +

State-vector 128 byte, depth reached 12, errors: 0
    10 states, stored
     0 states, matched
    10 transitions (= stored+matched)
     3 atomic steps
hash conflicts: 0 (resolved)
```

A continuación se explican las líneas mostradas en la salida de la ejecución, las cuales tienen importancia durante el proceso de verificación:

Full statespace search for: Indica el tipo de búsqueda realizada en el espacio de estados. Por defecto el tipo de búsqueda es la exhaustiva (full), es decir, chequea todos los posibles estados generados.

acceptance cycles + (selected): El signo + indica que se buscaron presencias de ciclos de no progreso. Esto se utiliza para verificar lazos de repetición específicos (como para un "repita para" o un "repita mientras"). De esta manera se chequea la existencia de lazos infinitos, que impedirían la finalización de la ejecución.

invalid endstates +: El signo + indica que se chequearon estados finales inválidos, es decir, ausencia o presencia de abrazos mortales.

State-vector 128 byte, depth reached 12, errors: 0 : La descripción completa del sistema global requirió 128 byte en memoria (por estado), y el camino de búsqueda más largo fue de 12 transiciones desde la raíz (estado inicial o INIT). En la búsqueda se encontraron 0 errores. Dado que el tipo de búsqueda realizado fue el exhaustivo, implica que se verifican cada uno de los estados generados en el proceso de ejecución. Al desarrollar tal ejecución, la búsqueda no obtuvo errores para alcanzar dichos estados.

0 states, matched: Esta línea indica que en ningún caso la búsqueda retornó un estado que se visitó previamente.

10 transitions (= stored+matched): Esto representa que un total de 10 transiciones fueron exploradas en la búsqueda, lo cual es un indicador de la cantidad de procesamiento que ha sido desarrollado para completar la verificación.

hash conflicts: 0 (resolved): Esta línea indica que en ningún caso el esquema de hashing que se uso para la verificación de los estados encontró conflictos o colisiones.

Así, no se encontraron ningún tipo de conflictos, por lo que no se generaron estados finales inválidos. Los estados finales inválidos surgen cuando se generan lazos infinitos o abrazos mortales. De esta manera los criterios de correctitud fueron probados y no indicaron errores de diseño.

5. Conclusión

Para lograr el diseño del protocolo propuesto se realizó un análisis exhaustivo del problema de mantenimiento de la coherencia en las memorias cache distribuidas. Hasta ahora no se habían desarrollado diseños que se enfocaran en sistemas con memoria cache distribuida y memoria principal distribuida. El protocolo cumplió con las exigencias de correctitud y consistencia lógica requeridas en los protocolos en los sistemas distribuidos.

Principalmente, el protocolo propuesto se caracteriza por utilizar un método que usa directorios distribuidos en cada nodo del sistema, los cuales almacenan la información sobre los datos guardados en las memorias cache, incluyendo sus estados actuales. Por medio de estos directorios distribuidos se logra manejar la coherencia de las memorias cache. Los directorios usan como estructura de datos dinámica un Árbol Binario Balanceado.

Los directorios distribuidos le permiten a cada nodo gestionar la información referente al conjunto de datos que le pertenecen, sin incurrir en costos de comunicación innecesarios. Para ello, el protocolo se basa en un conjunto de reglas que establecen el manejo de la memoria principal y cache en cada nodo del sistema, así como los intercambios de información entre los nodos. Todo esto garantiza la coherencia de los datos almacenados en las diferentes memorias del sistema.

Se realizó una validación que permitió validar dos criterios: el primero denominado *completitud*, para lo cual se utilizó por un lado un algoritmo exhaustivo, y por otro lado un análisis gráfico, dando como resultado un diseño que cumple con las exigencias de *completitud*. El segundo criterio se llamo *correctitud*, que se evaluó a través de una simulación y utilizando un verificador exhaustivo (ambos usando *Spin*), las cuales demostraron que el protocolo no contiene errores de diseño. Falta aun evaluar el rendimiento del protocolo bajo distintos esquemas de funcionamiento sobre plataformas reales.

Bibliografía

[1] Aguilar, J. y Leiss, E. A Cache Memory System based on a Dynamic/Adaptive Replacement Approach. *Revista Colombiana de Computación*. Vol. 4, N.1, pp. 7-20, 2003.

[2] Aguilar, J. y Leiss E. Introducción a la Computación Paralela. *Editorial Venezolana, Universidad de Los Andes,* Septiembre 2004.

[3] Aguilar, J. y Leiss, E. An Adaptive Coherence-Replacement Protocol for Web Proxy Cache Systems. *Revista Computación y Sistemas*, Vol. VIII. No 1, 2004.

[4] Alexander, T. y Kedem, G. Distributed Prefetch-buffer/Cache Design for High Performance Memory Systems, Technical Report. *Duke University*, 1996.

[5] Archival, J. y Baer, J. An economical Solution to the Cache Coherence Problem, Technical Report. *Department of Computer Science. University of Washington.* Seattle. 1984.

[6] Censier, L. y Feautrier, A New Solution to Coherence Problems in Multicache Systems. *IEEE, Trans. Comput,* vol. C(27), pp, 1112-1118, Dec. 1978.

[7] Chandra, S. Richard, B. y Larus, J. Language support for writing memory coherence protocols. *University of Wisconsin,* Madison, WI, United States. 1996.

[8] El-Ghazawi, T. Introduction to Cache and Distributed Memory Concepts, Technical Report. *The George Washington University.* Estados Unidos. 2002.

[9] Heinrich, M. The Performance And Scalability of Distributed Shared Memory Cache Coherence Protocols. PhD Dissertation. *Department of Electrical Engineering. Stanford University,* 1998.

[10] Hennessy, J. Heinrich, M. y Gupta, A. Cache Coherent Distributed Shared Memory: Perspectives on Its Development and Future Challenges. *Proceedings of the IEEE*, vol. 87, NO. 3. 1999.

[11] Shen, X. Arvind y Rudolph, L. CACHET: An Adaptive Cache Coherence Protocol for Distributed Shared-Memory. *Technical Report. Massachusetts Institute of Technology.* Proceedings of the 13th ACM-SIGARCH International Conference on Super Computing. Rhody, Greece, 1999.

[12] Shen, X. Arvind y Rudolph, L. Commit-Reconcile & Fence (CRF: A New Memory Model for Architects and Compiler Writers). Technical Report. *Massachusetts Institute of Technology*, 1999.

[13] Vega, M. Martín, R. Zarallo, F. Sánchez, J. y Gómez, J. SMPCache: Simulador de Sistemas de Memoria Caché en Multiprocesadores Simétricos. XI Jornadas de Paralelismo. Granada, 2000.

[14] Spin On line. Disponible en: http://spinroot.com/spin/whatispin.html.

Indice de autores / Index of Authors

Abalde, Carlos, 893
Acuña, Silvia T., 1059
Aguilar Castro, Jose L., 1123
Aguirre, Nazareno, 1049
Alarcón, Rosa, 195
Albernaz Amaral, Leonardo, 611
Alor Hernandez, Giner, 413
Alvares, Luís Otávio, 129
Alvarez, Gloria Inés, 239
Aniorte, Philippe, 955
Aráoz, Julián, 845
Arboleda, Hugo Fernando, 1071
Arcila Guzmán, Olmedo, 251
Ariza Acevedo, Pedro, 413
Ariza, Oscar, 345
Armero, Stiven, 465
Augusto Bezerra, Eduardo, 611
Ayala-Rincon, Mauricio, 41, 699

Bañón Pinar, José María, 251
Baeza-Yates, Ricardo, 29, 1103
Barán, Benjamín, 587
Barbosa, Claudia Andrea, 321
Barrera, Jorge, 121
Barrios N., Juan Manuel, 795
Bax, Marcello, 1027
Belchior, Arnaldo, 965
Benaderet, David, 97
Bernardini, Flavia Cristina, 151
Bigonha, Mariza, 881
Bigonha, Roberto, 881
Bittencourt, Ana C., 87
Blanco, Eduardo, 561, 599
Branch, John, 285
Bustacara Medina, César Julio, 355

Cámara Chávez, Guillermo, 11
Cabrero, David, 711, 893
Cadenas, Luis, 477
Caetano, César Augusto Cardoso, 73
Campana, Patrícia Targon, 73
Cano, Antonio, 239
Cardinale, Yudith, 599
Cardoso, Marcos Borba, 273
Carrillo Ramos, Angela Cristina, 367
Casallas, Rubby, 977
Castañeda Marin, Hernando, 549
Castelan Vega, Lucio Daniel, 413
Castro, Carlos, 19
Cayssials, Ricardo, 405
Cernuzzi, Luca, 635
Cerquera, Camilo Andrés, 465
Chavarriaga, Jaime Alberto, 1071

Cobos, Carlos, 203
Collazos, César, 195
Cordero, Ricardo D., 1059
Correal, Dario, 977
Costa, Antônio Carlos da Rocha, 273
Cruz, Harold, 525
Czekster, Ricardo, 771

D'Amico, Maria Belen, 833
Díaz Frías, Juan Francisco, 723
da Silva, Sergio Roberto, 759
Dai, Xiang, 537
Dantas de Menezes, Carlos Eduardo, 79
de Albuquerque Araújo, Arnaldo, 11
De Moya Amarís, Marly Esther, 175
De Oliveira, Jesus, 599
de Oliveira, Roberto, 163
Delgado, Alberto, 735
Deville, Yves, 623
Diaz, Idanis, 285
Diaz, Mario Fernando, 465
Diosa, Henry Alberto, 723
do Nascimento Salvador, Laís, 79
Doñate, Pedro, 833
Dorado, Andres, 297
Durscki, Roberto, 1001

Fabri, José Augusto, 1001
Fernández, Elena, 845
Fernandez Giangreco, José Manuel, 587
Fernandez, Alejandro, 227
Ferraz, Sandro Aparecido, 73
Ferro, Mariza, 109
Figueira, Carlos, 561
Figueroa, Christian, 19
Figueroa, Pablo, 345
Figueroa, Saritha G., 1059
Fiuri, Ariel Marcelo, 659
Frédérick, Seyler, 955
Freire, José Luis, 647

Gacitua, Veronica, 501
Gançarski, Stephane, 489
Gaona Cuevas, Carlos Mauricio, 723
García, Pedro, 239
Garcia, Guillermo, 187
Garcia, Rafael, 747
Gensel, Jérôme, 367
Giambiagi, Pablo, 905
Giraldo, Jorge, 809
Goldchleger, Andrei, 1083
Goncalves Da Silva, Marlene, 425
González Guerrero, Enrique, 355

Grimán, Anna C., 989
Guerrero, Luis, 195
Guerrero, Mario Fernando, 465
Gulías, Víctor M., 711, 893
Gutierrez, Claudio, 795
Guzmán, Liliana, 1015

Henriques, Pedro Rangel, 783
Hernández Hoyos, Marcela, 333
Hernández, Emilio, 477, 561
Herrera Cateriano, Patricia Shirley, 261
Hincapié, Roberto, 121

Iglezias, Leandro, 771
Iorio, Vladimir, 881
Irazabal, Jeronimo, 671
Izquierdo, Ebroul, 297

Janauskas, Ana, 97
Jeronymo, Marco Aurélio, 759
Jorge, J. Santiago, 711

Kinoshita, Jorge, 79
Kon, Fabio, 1083

Lazilha, Fabrício Ricardo, 215
León, Claudia, 489
Leal, Santiago, 345
Leiss, Ernst, 537
Librelotto, Giovani Rubert, 783
Lima, José Valdeni de, 215
Lladó, Catalina M., 869
Lopez, Nicolas, 977
Luna, Julio, 445

Müller, Gonzalo, 3
Magnani, Kristian, 881
Manssour, Isabel, 771
Marín Raventós, Gabriela, 453
Marquez, Mayerling, 393
Martin, Hervé, 367
Martinez Lopez, Pablo Ernesto, 671
Martinez, Cristian, 309
Martinez, Gabriel, 345
Mateus, Sandra, 285
Mauricio, David, 857
Menasalvas, Ernestina, 141
Mendoza, Luis E., 989
Mendoza, Martha, 821
Meneses, Rafael, 19
Mesa, Sergio Iván, 333
Meza, Oscar, 845
Millán, Socorro, 141
Miranda, Rodrigo, 41
Molinelli, José María, 647
Monard, Maria Carolina, 109, 151
Monfroy, Eric, 19
Montes, Pablo, 929
Montoya, Edwin, 809
Moraes, Silvia M.W., 87
Morales, Heidy Paola, 321
Morali, Ayse, 573
Moreno Cadavid, Julián, 59
Moreno, Jorge, 821
Muñoz, Diana, 525
Muñoz, Eduardo, 513
Muñoz, Leydy, 203

Naacke, Hubert, 489
Naranjo, Roberto, 821
Naso, Federico, 227
Niño Vasquéz, Luis Fernando, 175
Niño, Marcos, 747
Nonato, Luis Gustavo, 261
Novaira, María Marta, 1049

Ocampo, Ernesto, 97
Olarte, Carlos, 687, 735
Oliveira Junior, Edson Alves de, 215
Oliveira, Elisamara de, 73
Oliveira, Fabiola, 881
Olmedo Aguirre, Jose Oscar, 413
Ordínez, Leonardo, 833
Orozco, Javier, 833
Orozco, Javier Dario, 405
Ospina, Gloria, 809
Ovalle Carranza, Demetrio Arturo, 59

Pérez, Gilberto, 647
Pérez, Maria, 989
Parreiras, Fernando, 1027
Paulovich, Fernando Vieira, 943
Peña, Jorge, 121
Pedrycz, Witold, 297
Pereira, Wilmer, 187
Perez, Jorge Andres, 735
Permigiani, Sonia, 1049
Perotto, Filipo, 129
Poblete, Barbara, 1103
Pow-Sang Portillo, José Antonio, 1037
Prestes, Diego Galho, 273
Prieto, Flavio, 285
Puigjaner, Ramón, 869

Quesada, Luis, 623

Ramalho, José Carlos, 783
Ramirez, Cástulo, 809
Reiser, Renata Hax Sander, 273
Rivas-Suarez, Robinson, 513
Rivera, Luis, 857
Rocco, Claudio, 3
Rocha, Pascale, 965
Rodriguez Graterol, Wladimir, 549

Ron, Gonzalo, 187
Rosselló, Jerónia, 869
Rueda, Camilo, 687, 735
Ruiz, José, 239
Ruiz, Maryem, 809
Rukoz, Martha, 489
Rusu, Cristian, 51
Rusu, Virginia, 51

Salazar, Jaime, 203
Salinger, Alejandro, 29
Sanchez, German, 285
Santana, Thomas, 699
Santini, Pablo, 489
Santos, Rodrigo, 833
Schneck de Paula Pessôa, Marcelo, 1001
Schneider, Gerardo, 905
Sierra, Luz Marina, 821
Smith, Connie U., 869
Spínola, Mauro, 1001
Steinmacher, Igor Fábio, 215
Suárez, Ascánder, 917
Sumoza Matos, Rodolfo L., 1123

Takahashi, Silvia, 929
Teixeira, Otávio, 163
Timarán Pereira, Ricardo, 465
Tineo, Leonid, 425, 435, 445
Tolman, Camron, 379
Toro, Nilka, 513
Trindade, André, 1001

Uriza C., Luis Felipe, 333
Urribarrí, Wendi C., 917
Urriza, José Manuel, 405

Valbuena, Gustavo, 345
Valderruten, Alberto, 711, 893
Vallejos, Reinaldo, 501
Van Roy, Peter, 623
Varela, Carlos, 379, 573
Varela, Leonardo, 573
Vargas Del Valle, Manuel F., 453
Vicente, Erick, 857
Villanova-Oliver, Marlène, 367
Villapol, Maria E., 393
Visconti, Marcello, 1015

Yriarte, Vicente, 917

Zambonelli, Franco, 635
Zambrano, Jossie, 393
Zapata, Alejandra, 501
Zorrilla, Marta, 141
Zuluaga, María Alejandra, 333

Indice de claves / Index of keywors

$\rho_{arq} - calculus$, 723
3D Atlas of Human Kidney, 345
3D Educational Material, 345
3D Visualization in Medicine, 345
3D reconstruction, 285

Adaptation, 367
Adoption Process, 1015
Agent Modeling abstractions, 635
Agent-based computing, 635
Algebraic Model, 671
Algorithms, 309
Alloy, 1049
Anticipation, 215
Application Integration, 977
Approximate string matching, 41
Arc Routing Problems, 845
Artificial Intelligence, 109, 121
Artificial Neural Networks, 59, 87
AS/NZS 4360, 965
Asistentes Personales Digitales, 393
ASM, 881
Association Rules, 453
Association Task., 465
Atherosclerotic Plaque Characterization, 333
AUML, 635
Automated Text Categorization., 87
Automatic Interface Generation, 759

Biodiversity Informatics, 453
Bioinformatics, 97
Biological Systems, 687
Bisimulation, 659
Bluetooth, 393
Bounding Volumes, 251

Cache Memory, 1123
Catalog, 795
Class Diagrams., 203
Classification, 857
Clustering, 297, 857, 1103
CMMI, 965
Cognitive agent, 129
Coherence Protocols, 1123
Collaborative Activities, 195
Collaborative Systems, 203
Collision Detection, 251
Colombian Electric Energy Market, 59
Color Images, 355
Color Models, 355
Combinatorial Optimization, 845
Combining Classifiers, 151
Compilation with Distributed Libraries, 599
Compiler Techniques, 869
Complex Queries Resolution, 97
Component Metamodel, 955
Component-Based Development, 943
Components, 723
Computational Geometry, 251
Computational Grids, 561
Computational Security, 549
Computed Tomography, 333
Computer Algebra, 647
Computer Graphics, 261
Computer Support for Cooperative Learning, 203
Concurrent Constraint Programming, 687
Concurrent Process Calculus, 687
Concurrent Programming, 623
Congruence, 659
Constraint Programming, 19, 623, 735
Constraint Solving, 19
Constructivist artificial intelligence, 129
Content Management, 1027
Content-Based Image Retrieval, 513
Contexts of development, 1071
Convergence, 525
Cooperation, 215
Coordination models, 573
Coq system, 647
Critical Pairs, 699
CRM, 413
Cryptographic, 549
Cutting Tools, 771

Data Base Management Systems, 465
Data Grid, 477
Data mining, 141
Database query processing, 425
Decision Trees, 3
Design patterns, 893
Detection of contours, 73
Development team, 1059
Digital image processing, 73
Digital Images Processing, 321
Digital Libraries, 809
Discrete formalism models, 611
Distributed Computing, 1083
Distributed Execution, 599
Distributed Systems, 489, 723, 881, 1123
DM, 405
Domain Modeling, 1027
Domain Specific Languages, 671
DSL, 671
Dublin Core, 809
Dynamic programming., 747
Dynamic Reconfiguration, 1049

E-Business Models, 821
E-Commerce, 413, 821
E-Communities, 821
E-Learning, 141
ECommerce on Developing Countries, 821
Effort Estimation, 1037
Electronic Commerce, 573
Ellipsoide, 251
EMO, 187
End-User Extensible Applications, 759
Engineering, 1071
Environmental Monitoring.n, 51
Event Middleware, 977
Evolutionary Algorithms, 587
Evolving Algorithms, 187
Exact Algorithm, 251

Fault-Tolerance, 379
Feature Description Diagrams FD, 671
Feature Selection, 297
Filtering Algorithms, 623
Flexibilization, 215
Formal Methods, 711, 917, 943
Formal methods, 647
Formal Models, 735
Formal Specication, 917
Free Software, 79
Functional Programming, 711
Fuzzy ART, 11
Fuzzy Language, 445
Fuzzy Logic Expert Systems, 59

Gaia, 635
Game Theory, 163
Gaze detection, 73
Genetic Algorithms, 163, 285
Genomic Data Repository, 97
Geographic Information System, 175
Geographic Information Systems, 453
Global Software Development, 977
Grammatical inference, 239
Graphs, 783
Grasp, 857
Grid Computing, 1083
Grids Platforms, 599
Groupware Services, 227
Groupware Services Integration, 227

Hardware Verification, 699
Heuristic Algorithms, 845
High Performance Java Applications, 599
Horizontal Quantification, 435
Human capabilities, 1059
Hybrid Volume Rendering, 261

ICP, 285
IEEE STD 1540-2001, 965
Image Classification, 297
Image Moments, 333
Inductive Logic Programming, 109
Information Retrieval, 413
Information System, 955
Instant messaging, 537
Integration, 955
integrity, 537
Integrity Constraints, 489
Intelligent Software Agents, 59
Intelligent Systems, 59
Interchange Format, 869
Intermediate Representation Language, 881
Interoperability, 955
Intranet, 795
Invariant pattern recognition, 11
Inverted indices, 29
Investigative Strategy, 1071
Iterative-Incremental Lifecycle, 1037

JAI, 321
Java, 893
Java cards, 905
Javadoc Comments, 929

K-Means, 857
K-Means Clustering, 355
Knowledge Discovery in Databases, 465
Knowledge representation, 1027
Kohonen, 175

Lambda-Switch, 687

Machine Learning, 79, 109
Management of Distributed Libraries, 599
Market Domain, 1001
Marketing Strategies, 821
Measurement, 1015
Measurement Extraction, 771
Medical Illustration, 345
Medical Imaging, 771
Memory analysis, 905
Merging, 29
Message Passing, 623
Meta Heuristic, 857
Metadata, 795, 809
Metaheuristics, 309
Method of Investigation, 1071
Method overload, 893
Mobile Agents, 379
Mobile Work, 195
Mozart, 735
MPEG-7, 297
Multi Databases, 489
Multi-Agent Systems, 97, 367, 573, 881
Multicast Routing, 501
Multilevel Thresholding, 355

Multimedia Information Retrieval, 513
Multiobjective Optimization, 187, 587
Multiple search, 29

Naming Services, 379
Natural Language Processing, 79
Network Communications, 501
Neural Gas, 175
Neural Networks, 175
Non-determinism, 273
NTCC, 687

Object Oriented Programming, 203
Object-Oriented Database Language, 445
Object-Z specification language, 445
Online Auctions, 573
Ontologies, 477
Ontology, 783
Open Systems, 833
Operational Semantics, 723
Optimization methods, 121
Otsu's Method, 355
Oz, 19

Parallel and Distributed Systems, 561
Parallel Computing, 1083
Parallel program testing and debugging, 611
Parallelism, 587
Particle swarm optimization, 121
Patterns, 227
People/role fit metric, 1059
Performance Evaluation, 561
Performance Model, 869
Performance Tuning, 561
PMBOK, 965
Potential Distribution of Species, 453
Power-Aware, 833
Prisoner's Dilemma, 163
Probability., 659
Procedural Language, 425
Program Comprehension Workbenches, 929
Program Derivation, 917
Program extraction, 647
Program Renement, 917
Program Transformation, 917
Programming Environments, 273
Programming Languages, 735
Projection histograms, 11
Proof, 659
Proof Assistants, 699
Proof-checker, 647
Pseudoknots, 747
Pupil contour detection, 73
Python Programming Language, 273

Quality Model, 989
Quality of Service, 501
Queries, 1103
Query Languages, 425
Query Processing, 435
Query Routing, 367
Queueing Network Model, 869

Radial Basis Functions, 51
Range images, 285
RDF, 809
Real-Time, 833
Real-World Applications, 711
Registration, 285
Reinforcement learning, 121
Relevance Feedback, 413
Residual Finite State Automata (RFSA), 239
Reusability, 599
Reuse, 955
Rewriting, 699
Rewriting-Logic, 699
Risk Management, 965
RM, 405
RNA, 747
Role, 1059
Rule, 659
RUP, 965

Schedulability Test, 405
Search Engines, 513, 1103
Secondary structure prediction, 747
security, 537
Segment Market, 1001
Selection Method, 163
Self Organizing Map (SOM), 175
Semantic Grid, 477
Semantic Search, 795
Semantic Web, 783, 795
Semiotic Model, 759
Services, 809
Set operations, 29
Similarity Measures, 3
SIP, 525
Software Architectures, 723, 1049
Software Development, 943
Software Engineering, 977, 1027, 1071
Software Engineering Experimentation, 1037
Software Factory, 1001
Software Line-Product, 1001
Software Performance Engineering, 869
Software Practice, 1015
Software process, 1059
Software Testing, 977
Software Verication, 711
Soporte Didáctico, 393
Source Code Representation, 929
Spam, 87
Spatial Data Mining, 453

SPEA2, 187
SQLf, 425, 435
Static analysis, 905
Statistical Processing of Language, 79
Stochastic Automata Networks, 611
Stochastic Context-Free Grammars (SCFG), 747
suffix arrays, 41
suffix trees, 41
Surface Modelling, 51
Symbolic Ensembles, 151
Symbolic Machine Learning, 151
System Reliability Assessment, 3

Team Algorithm., 587
Theorem Provers, 711
Thresholding, 355
Tool Interoperability, 869
Topic Maps, 783
Topic Maps Visualization, 783
Transaction Systems, 489
Transfer Function, 771
Transition systems, 659
Traveling Salesman Problem, 163

Ubiquitous Computing, 367
UML, 203
UML2.0, 723
Universal Actors, 379
Unstructured Meshes, 261
Usability and Sociability, 821
Use Cases, 1037
User Assistance, 97
User Assistant Agents, 97

Validation and Verification, 1049
Vascular image processing, 333
Vessel Segmentation, 333
Visual Editor, 321
Visual Languages, 273
Visual Programming, 321
Visualization, 771
Voice portal servers, 525
Voice portals, 525
VoiceXML, 525
Volume Visualization, 261

Web Mining, 1103
Web search engines, 29
Web Service, 989
Webhouse, 141
Website Improvement, 1103
Website Mining, 1103
Workflow, 215

XML, 869, 929
XSL Transformations, 929