CLEI

Panel 96

Memorias - 2

Pectoral Antropozoomorfo - Museo del Oro

XXII Conferencia Latinoamericana de Informática

Santafé de Bogotá

Colombia

Junio 3 - 7, 1996

ACIS

CLEI
CENTRO LATINOAMERICANO DE
ESTUDIOS EN INFORMATICA

universidad de los andes

PEDECIBA
INFORMATICA

XXII CONFERENCIA LATINOAMERICANA DE INFORMÁTICA
CLEI PANEL 96

Santafé de Bogotá
Colombia
Junio 3 - 7, 1996

Memorias
Volumen 2

No. INV 2284

EDICIÓN
Rodrigo Cardoso
Universidad de los Andes

REALIZACIÓN
Asociación Colombiana de Ingenieros de Sistemas (ACIS)
Universidad de los Andes - Departamento de Ingeniería de Sistemas y Computación

Carátula : Pectoral antropozoomorfo
Museo del Oro
Banco de la República
Santafé de Bogotá, Colombia

Impresión : Centro de Publicaciones
Universidad de los Andes
Santafé de Bogotá, Colombia

Copias adicionales : Universidad de los Andes
Departamento de Ingeniería de Sistemas y Computación
A.A. 4976
Santafé de Bogotá, Colombia
Tel.: (57 1) 2866185
Fax: (57 1) 2840796

ACIS
A.A. 94334
Santafé de Bogotá, Colombia
Tel.: (57 1) 6161407
Fax: (57 1) 6161409

PREFACIO

La Conferencia Latinoamericana de Informática tiene lugar en Santafé de Bogotá, Colombia, nueve años después de haberse realizado por vez primera en este país. Esta edición - la vigésimo segunda - pone de relieve, como ya hace varios años, la madurez alcanzada por el Centro Latinoamericano de Estudios en Informática (CLEI) para concitar a la comunidad informática latinoamericana en un encuentro anual donde comunica y comparte sus logros científicos y tecnológicos.

Al CLEI Panel 96, la principal actividad académica de la conferencia, en la que autores de la región presentan a sus pares los resultados más recientes de su investigación, se sometieron 190 artículos. Estos fueron juzgados por un Comité de Programa internacional, compuesto por 25 académicos de Iberoamérica, Estados Unidos y Francia. Al final se seleccionaron 98 artículos, de los cuales 95 serán presentados en CLEI Panel 96 y hacen parte de estas Memorias. Adicionalmente, dentro de Panel 96, se expondrá el trabajo ganador del III Concurso CLEI - UNESCO de Tesis de Maestría en Informática.

Además de la presentación de artículos aceptados, dentro de la Conferencia se ofrecen 11 charlas y 6 cursillos tutoriales a cargo de prestigiosos representantes de la academia y de la industria informática. De esta forma, se brinda a los asistentes un panorama de la actualidad informática en temas como computación de alto desempeño, inteligencia artificial, redes, Internet, bases de datos y aplicaciones, entre otros.

La organización del evento ha sido una labor conjunta de la Asociación Colombiana de Ingenieros de Sistemas (ACIS) y del Departamento de Ingeniería de Sistemas y Computación de la Universidad de los Andes. Es para nosotros un honor servir de anfitriones a nuestros colegas y corresponder de esta manera la realización de anteriores conferencias, a la vez que traer, al gremio informático colombiano, un congreso internacional de las características de CLEI Panel 96.

El éxito de esta XXII Conferencia Latinoamericana de Informática ha dependido de la colaboración de muchas personas, movidas por el único interés de hacer más grande la informática en Latinoamérica. De las directivas de CLEI, con su labor de difusión en cada país participante; de los miembros de los Comités de Programa -tanto de Panel 96 como del Concurso de Tesis de Maestría- que además contaron con el apoyo de sus instituciones en la labor de evaluación; de colegas y asistentes de ACIS y de la Universidad de los Andes, con días de trabajo intenso y silencioso.

Por último, vaya un agradecimiento sincero a los muchos patrocinadores y auspiciadores de CLEI Panel 96. Con nuestra gratitud, la certidumbre de que de esta manera contribuimos de manera tangible al progreso de nuestros pueblos.

Rodrigo Cardoso
XXII Conferencia Latinoamericana de Informática CLEI Panel 96
Coordinador General

Santafé de Bogotá, junio de 1996

XXII Conferencia Latinoamericana de Informática
CLEI Panel 96
Comité de Programa

Presidente	:	José Tiberio Hernández Universidad de los Andes, Bogotá, Colombia
Coordinador	:	Rodrigo Cardoso Universidad de los Andes, Bogotá, Colombia

Comité de evaluación:

Jorge Santos	Universidad Nacional del Sur, Argentina
Gustavo Rossi	Universidad Nacional de La Plata, Argentina
Claudia Medeiros	Universidade Estadual de Campinas, Brasil
Carlos Heuser	Universidade Federal do Rio Grande do Sul, Brasil
Ana Carolina Salgado	Universidade Federal do Pernambuco, Brasil
Vera L.S. Lima	Pontificia Universitate Católica do Rio Grande do Sul, Brasil
Ricardo Baeza-Yates	Universidad de Chile, Chile
Leopoldo Bertossi	Pontificia Universidad Católica, Chile
Christian Trefttz	Escuela de Administración y Finanzas EAFIT, Colombia
Juan Francisco Díaz	Universidad del Valle, Colombia
Camilo Rueda	Universidad Javeriana de Cali, Colombia
José Tiberio Hernández	Universidad de los Andes, Bogotá, Colombia
Claudia Roncancio	INPG Grenoble, Francia
Ignacio Trejos	Instituto Tecnológico de Costa Rica, Costa Rica
Ramon Puigjaner	Universidad de Islas Balèares, España
José Neira	Universidad de Zaragoza, España
Isidro Ramos	Universidad Politécnica de Valencia, España
Marcelo Mejía	Instituto Tecnológico Autónomo de México, México
Luis A. Trejo	Instituto Tecnológico de Monterrey, México
Manuel Ibarra	CINVESTAV, México
Juan Vicente Echagüe	Universidad de la República, Uruguay
Ernst Leiss	University of Houston, USA
Alfredo Matteo	Universidad Central de Venezuela, Venezuela
Ascánder Suarez	Universidad Simón Bolivar, Venezuela
Victor Theotkisto	Universidad Metropolitana, Venezuela

LISTA DE EVALUADORES

La labor de evaluación de cada uno de los miembros del Comité de Programa fue apoyada por colegas de sus instituciones, cuyos nombres se incluyen a continuación, de acuerdo a la información suministrada por los miembros del Comité a la Organización.

Gustavo Rossi. LIFIA- Universidad Nacional de la Plata, Argentina
Nelida Etchebest - LIFIA, UNLP, Arg.
Silvia Gordillo - LIFIA, UNLP, Arg.

Jorge Santos - Universidad Nacional del Sur, Argentina
Edgardo Ferro - UNS, Arg.
Ricardo Cayssials - UNS, Arg.

Claudia M Bauzer Medeiros - Universidade Estadual de Campinas, Brasil
Mariano Cilia - UEC, Bra.
Cecilia Rubira - UEC, Bra.
Arnaldo Moura - UEC, Bra.
Juliano Lopes de Oliveira - UEC, Bra.
Heloisa Vieira da Rocha - UEC, Bra.
Nevton Cereja - UEC, Bra.
Nelson Fonseca - UEC, Bra.
Ana Cristina Vieira de Melo - UEC, Bra.
Eliane Martins - UEC, Bra.
Luiz Eduardo Buzato - UEC, Bra.
Ariadne Rizzoni Carvalho - UEC, Bra.
Fabio Nogueira de Lucena - UEC, Bra.

Vera L.Strube de Lima - Inst. de Informatica, PUCRS, Brasil
Afonso Inacio Orth - PUCRS, Bra.
Bertilo Frederico Becker - PUCRS, Bra.
Duncan D. A. Ruiz - PUCRS, Bra.
Flavio Moreira de Oliveira - PUCRS, Bra.
Karin Becker - PUCRS, Bra.
Ney Laert Vilar Calazans - PUCRS, Bra.
Vera Lucia Strube de Lima - PUCRS, Bra.
Antonio Carlos da Rocha Costa - PUCRS, Bra.

Ana Carolina Salgado - Universidade Federal do Pernambuco, Brasil
Alejandro Cesar Frery - UFPE, Bra.
Carlos Andre Ferraz - UFPE, Bra.
Celso Roberto Perez - UFPE, Bra.
Edson Carvalho Filho - UFPE, Bra.
Fabio Queda Bueno da Silva - UFPE, Bra.
Germano Crispim Vasconcelos - UFPE, Bra.
Fernando da Fonseca de Sousa - UFPE, Bra.
Jaelson Brelaz de Castro - UFPE, Bra.
Judith Kelner - UFPE, Bra.
Roberto Souto Maior Barros - UFPE, Bra.

Leopoldo Bertossi - Pontificia Universidad Católica, Chile
Alvaro Campos - PUC, Chi.
Carlos Hurtado - PUC, Chi.
Fritz Knabe - PUC, Chi.
Yadran Eterovic - PUC, Chi.
Miguel Nussbaum - PUC, Chi.

Ricardo Baeza-Yates - Universidad de Chile, Chile

Tiberio Hernandez - Universidad de los Andes, Colombia
Jorge Villalobos - Uniandes, Col.
Silvia Takahashi - Uniandes, Col.
Germán Bravo - Uniandes, Col.
Carlos Varela - University of Illinois, USA
Alejandro Quintero - Uniandes, Col.
Dario Correal - Uniandes, Col.
Alvaro Galvis - Uniandes, Col.
Rodrigo Cardoso - Uniandes, Col.
Rodrigo López - Uniandes, Col.
María Consuelo Franky - Uniandes, Col.
Olga Lucía Giraldo - Uniandes, Col.
Claudia Jiménez - Uniandes, Col.

José Abásolo - Uniandes, Col.
Victor M. Toro - Uniandes, Col.
Martín Soto - Uniandes, Col.
Pablo Figueroa - Uniandes, Col.
Hernán Aponte - Uniandes, Col.
Olga Mariño - Uniandes, Col.
Harold Castro - Uniandes, Col.
Francisco Rueda - Uniandes, Col.
Ernesto Lleras - Uniandes, Col.

Camilo Rueda - Universidad Javeriana de Cali, Colombia
Andres Jaramillo - UJC, Col.
Gabriel Tamura - UJC, Col.
Eugenio Tamura - UJC, Col.

Juan Francisco Diaz - Universidad del Valle, Colombia
Fernando Machuca - UniValle, Col.
Jose Banón - UniValle, Col.
Gloria Toro - UniValle, Col.
Maria Eugenia Valencia - UniValle, Col.
Hector Angulo - UniValle, Col.

Christian Trefftz Gómez - EAFIT, Colombia
Martha Eugenia Alvarez - EAFIT, Col.
Rodrigo Carvajal - EAFIT, Col.
Monica Henao Calad - EAFIT, Col.
Juan Guillermo Lalinde Pulido - EAFIT, Col.
Felix Londonho - EAFIT, Col.
Bertha Alicia Solorzano - EAFIT, Col.
Helmuth Trefftz Gomez - EAFIT, Col.
Claudia Zea Restrepo - EAFIT, Col.

Ignacio Trejos - Depto. Computacion, Instituto Tecnologico de Costa Rica, Costa Rica
Freddy Abarca - ITCR, CR.
Jose E. Araya - ITCR, CR.
Allan Barrantes - ITCR, CR.
Rodrigo Bogarin - ITCR, CR.
Rodolfo Calvo - ITCR, CR.
Carlos Gonzalez - ITCR, CR.
Jose Helo - ITCR, CR.
Erick Mata - ITCR, CR.
Anibal Mayorga - ITCR, CR.
Liliana Sancho - ITCR, CR.
Edison Campos de Faria - UCR, CR.
Juan Cordoba - ITCR, CR.

Isidro Ramos - Universidad Publica de Valencia, España
Roberto Vivo - UPV, Esp.
Vicente Botti - UPV, Esp.
Francisco Casacuberta - UPV, Esp.
Oscar Pastor - UPV, Esp.
Javier Oliver - UPV, Esp.
Hilario Canos - UPV, Esp.
Maria Alpuente - UPV, Esp.

Ramon Puigjaner - Universidad Islas Baleares, España
Tomasa Calvo - ISB, Esp.
Gabriel Fontanet - ISB, Esp.
Maria Luisa Santamaria - ISB, Esp.
Sebastia Galmes - ISB, Esp.
Isabel Perello - ISB, Esp.
Francisco Bastida - ISB, Esp.

Jose Neira - Universidad de Zaragoza, España
Jose Angel Banyares - Uzar, Esp.
Jose Maria Martinez - Uzar, Esp.
Francisco Seron - Uzar, Esp.
Jose Luis Villarroel - Uzar, Esp.
Carlos Sagues Blazquiz - Uzar, Esp.
Joaquin Ezpeleta - Uzar, Esp.
Juan Domingo Tardos Solano - Uzar, Esp.
Luis Montano Gella - Uzar, Esp.
Jose Luis Briz - Uzar, Esp.

Claudia Roncancio - INPGrenoble (Francia)
Rubby Casallas -INPG, Fra.
Jose Celso Freire -INPG, Fra.

Luis A. Trejo R. - Instituto Tecnológico de Monterrey CEM, México
Alfonso Esparza B. - ITESM-CEM, Mex..
Eduardo Garcia - ITESM-CEM, Mex..
Roberto Gómez - ITESM-CEM, Mex.
Monica López - ITESM-CEM, Mex..
Carlos Lucatero - ITESM-CEM, Mex..
Carlos Martinez. - ITESM-CEM, Mex..
Ariel Ortiz - ITESM-CEM, Mex..
Rafain Rodríguez - ITESM-CEM, Mex..
Miguel Salais - ITESM-CEM, Mex..
Jesús Sanchez - ITESM-CEM, Mex..
Carlos Sandoval C. - ITESM-CEM, Mex..
Guillermo Sandoval - ITESM-CEM, Mex..
Luis A. Trejo - ITESM-CEM, Mex..
Jesús Vazquez - ITESM-CEM, Mex..
Hector Villa - ITESM-CEM, Mex..

Marcelo Mejia - Instituto Tecnológico Autónomo de México, México
Carlos Zozaya - ITAM, Mex.
Francisco Cervantes - ITAM, Mex.
Alfredo Weitzenfeld - ITAM, Mex.
Alfonso San Miguel - ITAM, Mex.
Armando Maldonado - ITAM, Mex.
Virginia Vergara - ITAM, Mex.
Ramon Rios - ITAM, Mex.
Jose Incera - ITAM, Mex.

Juan Manuel Ibarra - Grupo de Robotica y Vision del Cinvestav-IPN, México
Alejandro Malo Tamayo - Cinvestav-IPN, Mex.
Elena Gortcheva - Cinvestav-IPN, Mex.

Juan Vicente Echague - Universidad de la República, Uruguay
Pedro D'Argenio, TIOS, Univ. de Twente, Hol.
Delia Kesner, LRI, Univ. Paris Sud (Orsay), Fra.
Sergio Yovino, VERIMAG, IMAG, Fra.

Ernst Leiss - University of Houston, USA

Alfredo Matteo - Universidad Central de Venezuela, Venezuela
Alejandro Crema - UCV, Ven.
Francis Losavio - UCV, Ven.
Fernando Martinez - UCV, Ven.
Christiane Metzner - UCV, Ven.
M. Angelica Ovalles - USB, Ven.
Arturo Sanchez - UCV, Ven.
Nunzio Savino - USB, Ven.
Hugo Segovia - UCV, Ven.

Ascander Suarez - Universidad Simon Bolivar, Venezuela
Maria Eugenia Fuenmayor - USB, Ven.
Maria Blanca Ibáñez - USB, Ven.
Maruja Ortega - USB, Ven.
Juan Carlos Guzman - USB, Ven.
Alonso Marquez - USB, Ven.

Victor Theoktisto - Universidad Metropolitana, Venezuela
Adelaide Bianchini - UM, Ven.
Pilar Cuenca - UM, Ven.
Edna Ruckhaus - UM, Ven.
Antonieta Bosque - UM, Ven.

III Concurso de Tesis de Maestría CLEI - UNESCO

Coordinadora:

Patricia Corbo
UNESCO - ORCYT, Uruguay

Coordinadores asociados:

Juan V. Echagüe
Maria Urquhart
Alfredo Viola
Luis Petingi
Raul Ruggia

Comité de evaluación:

Carlos Alberto Heuser	Universidade Federal do Rio Grande do Sul, Brasil
Javier Pinto	Pontificia Universidad Católica de Chile, Chile
Christian Trefftz	Escuela de Administración y Finanzas EAFIT, Colombia
Jorge Villalobos	Universidad de los Andes, Colombia
Raul Monge	Universidad Tecnica Federico Santa Maria
Alejandro Casares	Pontificia Universidad Catolica de Ecuador, Ecuador
Lourdes Sanchez	Universidad Autónoma Metropolitana, México
Benjamin Baran	Universidad Nacional de Asunción, Paraguay
Alfredo Viola	Universidad de la Republica Oriental del Uruguay, Uruguay
Cristina Zoltan	Universidad Simon Bolivar, Venezuela

Asociación Colombiana de Ingenieros de Sistemas
ACIS

Junta directiva 1996 -1997

Presidente	:	Carlos H. Ardila
Vicepresidente	:	Álvaro Torres
Secretario	:	Rodrigo Querubín
Tesorero	:	Germán Villarreal
Vocal	:	Jorge E. Tarazona
Vocal	:	Víctor M. Toro
Vocal	:	Guillermo Santos
Revisor fiscal	:	Juan P. Pazos

CENTRO LATINOAMERICANO DE ESTUDIOS EN INFORMÁTICA

Presidente : Edgardo Broner
Universidad Simón Bolivar, Caracas, Venezuela

Secretario : David Fuller
Pontificia Universidad Católica de Chile, Santiago, Chile

Tesorero : Gerónimo Bellasai
Universidad Católica, Asunción, Paraguay

Representantes por países:

ARGENTINA	Jorge Santos	Universidad Nacional del Sur Bahia Blanca
BRASIL	Carlos Heuser	Universidade Federal do Rio Grande do Sul Porto Alegre
CHILE	Javier Cañas	Universidad Técnica Federico Santa María Valparaiso
COLOMBIA	Rodrigo Cardoso	Universidad de los Andes Bogotá
ECUADOR	Diego Andrade	Pontificia Universidad Católica Quito
MÉXICO	Veronica Mendoza	Instituto Tecnológico y de Estudios Superiores de Monterrey México D.F.
PARAGUAY	Geronimo Bellasai	Universidad Católica Asunción
URUGUAY	Marita Urquhart	Universidad de la Republica Montevideo
VENEZUELA	Oscar Meza	Universidad Simon Bolivar Caracas

TABLA DE CONTENIDO

Volumen 1

ORIENTACIÓN A OBJETOS I

An object-oriented architecture for decision support systems
Becker, Karin; Bodart, François 1

Uma ferramenta de apoio à metodologia OMT para integração de modelos
González, Marco; Orth, Afonso 14

Alternativas para implementar clases genéricas en C++
Jenkins, Marcelo; Villalón, Ricardo; Alvarado, Marco 28

Interface agents frameworks for object-oriented design
Losavio, Francis; Matteo, Alfredo 38

INGENIERÍA DE SOFTWARE I

Modelo de madurez para el proceso de aseguramiento de calidad de software (SQA)
Rojas, Patricio; Visconti, Marcello 51

Aplicação da teoria Fuzzy em requisitos de qualidade de software
Belchior, Arnaldo; Xexéo, Geraldo; da Rocha, Ana 63

Regras heurísticas para transformação automática de textos em diagramas de fluxo de dados
Sturm, Carolina; de Moraes, Rodrigo; Loh, Stanley 75

TRABAJO COOPERATIVO I

An implementation model for collaborative applications
Cortés, Mauricio; Mishra, Prateek 87

A flexible and adaptable virtual class in a tele-teaching environment
Hernández, Arturo 99

Modelo de un sistema de soporte a la negociación en grupo
Hernández, Marcela; Quintero, Alejandro 115

Un modelo de comunicación de soporte a la planificación flexible en un sistema de agentes cooperantes
Ramos, Fernando; Liñán, José L. 127

BASES DE DATOS I

Use of active rules in the schema evolution process of an object manager system
Moura, Ana Maria; Guimarães, Ricardo; Barros, Ricardo 139

SysKBZ+, um protótipo de SGBDOO baseado no modelo KBZ+
Toranzo, Marco; da Fonseca, Fernando; Vasconcelos, Alexandre; Bezerra, Hednilson; Monteiro, Fred; Soares, Breno 152

La riqueza formal presente en el vínculo entre un modelo de bases de objetos y un modelo de transacciones
Smith, Clara; Tau, Carlos 165

Las dependencias multivaluadas como reglas de inferencia
Millán, Marta; Fernández, María C. 177

ALGORITMOS I

Optimización de la base de reglas y de las funciones de membresía de un controlador difuso utilizando algoritmos genéticos
Mejía, Marcelo; Mejía, Antonio 191

A genetic algorithm for the forwarding Index/diameter of graphs
Barráez, Daniel; Domínguez, Ramón 203

Análisis de paralelización en un algoritmo genético para diseño de antenas
Anciano, Juan; Arráiz, Emely; Savino, Nunzio 214

A study of the fitness function in the genetic algorithm
Aguilar, José 226

PARALELISMO PRÁCTICO

Mapaná: una plataforma paralela para red de microcomputadores
Corzo, Fabio; Gómez, Rafael 238

Comparación de objetos complejos multiversión: una solución paralela
Rukoz, Marta; Jomier, Geneviève; Delahaye, Jean; Meléndez, Iraida 250

Analytical evaluation of the 2D-DCT using paralleling processing
Di Serio, Angela 262

Sequential and parallel computation strategies on coherence spaces
Schneider, Ruben; da Rocha, Antônio 276

INFORMÁTICA TEÓRICA

Un sistema de inferencia de tipos extendidos con información de terminación
Salas, Jorge; Suárez, Ascánder 288

Proof representation in type theory: state of the art
Muñoz, César 300

Tipos algebraicos de datos en fork álgebras
Martínez,Pablo; Baum, Gabriel 310

Transforming CRSs into TRSs - About elimination of the conditions -
Ayala, Mauricio 322

ALGORITMOS II

A unified view to pattern matching problems
Baeza-Yates, Ricardo 335

Virus for graph properties
Brito, Ma. Rosa; Delorme, Charles; Márquez, Luis E.; Ordaz, Oscar 347

Facets for matroidal-knapsack and knapsack problems
Aráoz, Julián; Maffioli, Francesco 359

El impacto del *cache* en dos algoritmos de ordenamiento
Arráiz, Emely; Curiel, Mariela; Savino, Nunzio 371

SISTEMA OPERACIONAL

A user-level parallel file I/O
Wang, Yili; Leiss, Ernst 383

User-mode DMA
Nava, Alberto; Kummerfeld, Bob 395

Las entradas/salidas en la computación de alto desempeño
Castro, Harold 405

Sistema auxiliar de arquivos baseado em disco ótico WORM para ambiente computacional distribuído
Santana, Regina; Santana, Marcos; Bezerra, Luiz 417

BASES DE DATOS II

Applying similarity vectors for the selection of reusable schemas
Ambrosio, Ana Paula; Georges, Fouad 430

Diseño e implantación de aplicaciones distribuidas sobre bases de datos relacionales: un enfoque orientado a objetos
Abásolo, José; Ortiz, Martha 442

Diseño e implantación de un optimizador de consultas para un sistema manejador de bases de datos orientado por objetos MIDBS
Hernández, Domingo 454

Descoberta de conhecimento aplicada à detecção de anomalias em bases de dados
Feldens, Miguel; Castilho, José 468

TRABAJO COOPERATIVO II

Negociación distribuida y semi-centralizada en sistemas multiagentes: dos modelos propuestos
Camargo, Francisco; Martínez, Rafael; Ramos, Fernando 482

Um ambiente de suporte à autoria cooperativa de hiperdocumentos
Dietrich, Elton; Valdeni, José 494

Protocolos de interacción de un sistema multi-agente como apoyo a la negociación y coordinación en grupo
Ucrós, Ma. Eugenia; Quintero, Alejandro 506

IMPLEMENTACIÓN DE LENGUAJES

Objects in a functional environment
Guzmán, Juan C.; Suárez, Ascánder 517

Una extensión de Caml-light para permitir definiciones de listas por comprensión
López, Rodrigo; Tamura, Gabriel 529

Type models: toward unifying concepts for language interoperability
Sánchez, Arturo 541

Herramienta de apoyo al desarrollo de compiladores en programación orientada por objetos
Bello, Humberto; Rodríguez, Rosseline; Tineo, Leodin 553

SISTEMAS DISTRIBUIDOS Y DE TIEMPO REAL

AgaPé-TR: una herramienta para simulación de planificación centralizada de procesos en tiempo real
Wainer, Gabriel 565

Sistemas de tiempo real en la industria, análisis del conjunto control/comunicaciones
Santos, Rodrigo; Masson, Favio; Zambón, Marcelo; Santos, Jorge 577

Diseño y aplicación de métricas de software a lenguajes de desarrollo de productos cliente/servidor
Ortiz, Carlos; Sánchez, Marcelo; Cañas, Javier; Visconti, Marcello 589

Usando una extensión al problema de los filósofos que cenan para estudiar sistemas de tiempo real duro
Wainer, Gabriel 602

Volumen 2

MULTIMEDIOS I

Authentication for multimedia documents
Chen, Fan; Leiss, Ernst 613

Estações multimídia de atendimento: uma abordagem ergonômica
Santos, Marizete; da Fonseca de Souza, Fernando; Salgado, Ana C. 625

Testing the usability of an hypermedia encyclopedia on data networks
Díaz, Javier; Harari, Ivana; Queiruga, Claudia 637

Um sistema de autoria para construção de tutores hipermidia
Rocha, Ana; Rabelo Jr., Alvaro; Campos, Gilda; Pavel, Paulo; Abreu, Mario 647

COMPUTACIÓN GRÁFICA

Métodos de impresión fiel para huellas dactilares
Drets, Gustavo; Curbelo, Rosario; Bergengruen, Olaf; Almansa, Andrés 659

A class of linear algorithms to process sets of segments
Navarro, Gonzalo; Baeza-Yates, Ricardo 671

Cálculo aproximado de diagramas de Voronoi utilizando autómatas celulares
Trefftz, Christian 683

Modelo para la simulación de fenómenos turbulentos para ambientes de realidad virtual
Barrero, Daniel; Hernández, José T. 689

PARALELISMO TEÓRICO

Marco formal para la especificación de implementaciones de la FFT sobre transputers
von Brand, Horst; Barra, Carlos 702

Partición de sistemas de ecuaciones para su resolución distribuida
Barán, Benjamín; Benítez, Diana; Ramos, Rodrigo 715

Semántica de sistemas reactivos. Un preorden entre procesos basado en la ST-bisimulación
Fialco, Gabriel; Echagüe, Juan 727

YALI - Uma extensão do modelo Linda para programação paralela em redes heterogêneas
Schwertner, Andréa; da Costa, Celso 739

ROBOTICA Y CONTROL

Desarrollo de un sistema de simulación y programación para el robot experimental SALVIATI
Fematt, Rafael; Juárez, Eliseo; Pámanes, Alfonso; Franco, Guillermo 751

Asistente de planificación de trayectorias libres de colisión
Godoy, Ma. Alejandra; Morales, Jorge 760

Estudios de algunos sistemas CACSD
Moreno, Rodrigo; Henández, José T. 772

REDES

Firewall monitoring
You, Jianyu; Leiss, Ernst 785

A firewall access definition language
You, Jianyu; Leiss, Ernst 797

Plataforma de análisis basada en modelos computacionales para simular BISDN a nivel ATM
Germinal, Isern ; Rivas, Rachele; Millán, Yrieslis 808

ORIENTACIÓN A OBJETOS II

A generic architecture for use-case based object-oriented design
Losavio, Francis; Matteo, Alfredo 821

Uma abordagem orientada a objetos para reutilização baseada em linguagens de domínio
Araujo, Flavio; Werner, Cláudia 833

A method for object-oriented user-interface design based on use-cases an multiagents models
Antoneti, Jenny; Losavio, Francis; Matteo, Alfredo 845

Un lenguaje de programación orientado a objetos activos
Pons, Claudia; Baum, Gabriel 857

MULTIMEDIOS II

Aplicando modelos de procesos de software al desarrollo de aplicaciones hipermedia
Montilva,Jonás 870

Diseño y consultas de aplicaciones hipermedias
Gordillo, Silvia; Díaz, Alicia 882

HSM - Modelagem semântica para aplicação hipertexto
Ladeira, Marcelo; Otsuka, Joice Lee; Valdeni, José; Silveira, Ricardo 894

ACAM - an active context-based authoring model
Kelner, Judith; Sadok, Djamel; Martins, Ricardo 907

INTELIGENCIA ARTIFICIAL

Complementing the space-time constraints paradigm with event facilities
Franco, José; Magalhães, Leo; Raposo, Alberto 922

Uso de deducciones de no viabilidad en el cálculo de arco-consistencia
Valencia, Frank; Rueda, Camilo 929

Un micromundo para la resolución de problemas de demostración en geometría
Luengo, Vanda 941

INGENIERÍA DE SOFTWARE II

Costos de la calidad y del mejoramiento del proceso de software
Concha, Mauricio; Visconti, Marcello *953*

DesProc 1.0. Una herramienta para descomposición formal funcional de procesos
Arias, Tatiana; Mora, Javier; Llamosa, Ricardo *967*

Un algoritmo para reestructurar programas procedurales
Cobo, Hernán; Mauco, Ma. Virginia *979*

A utilização de múltiplos métodos formais na análise de requisitos de software
dos Santos, Simone; da Silva, Fabio *991*

Experiência do GMC no desenvolvimento de software instrucional
Diverio, Tiarajú; Tavares, Beatriz *1004*

REDES NEURONALES

Recognizing matching among terms: a neural network approach
Mareco, Carlos; Paccanaro, Alberto *1012*

Diseño e implementación de redes híbridas neuro-difusas para el control de procesos industriales
Alvarez, Hernán; Ovalle, Demetrio *1021*

Red neuronal para reconocer minerales en una imagen digitalizada
Solar, Mauricio; Watkins, Francisco; Pérez, Patricio *1033*

Codificación de video digital basado en la estimación implícita de movimiento inter-imagen utilizando redes de neuronas
García, Eduardo *1045*

SISTEMAS DISTRIBUIDOS

Manejo de archivos replicados bajo transacciones distribuidas en el sistema DGDBM
Castro, Orlando; Franky, Ma.Consuelo *1057*

DS-LED: a distributed application for the simplex method
Herrero, Jorge; Paccanaro, Alberto *1069*

Una experiencia de programación distribuida utilizando Paradise
Mejía, Marcelo *1080*

Priority inversions in rate monotonic 8802/4: Analysis of the worst case
Cayssials, Ricardo; Orozco, Javier; Santos, Jorge; Ferro, Edgardo *1092*

Diagramación dinámica de tareas de tiempo real duro con constricciones de precedencia en sistemas distribuidos multitarea-multiprocesador
Cayssials, Ricardo; Orozco, Javier; Santos, Jorge; Ferro, Edgardo *1101*

PLANEACIÓN ESTRATÉGICA

El impacto de la tecnología de información en la empresa de los 90's.
Rueda, Francisco *1113*

III CONCURSO DE TESIS DE MAGISTER CLEI / UNESCO - PRIMER LUGAR

Un lenguaje de consultas sobre estructura y contenido en bases de datos de texto
Navarro, Gonzalo *1125*

Authentication for Multimedia Documents

Fan Chen and Ernst L. Leiss
Department of Computer Science
University of Houston
Houston Texas 77204-3475
coscel@cs.uh.edu

Abstract

We discuss problems of data security for multimedia documents. While many techniques have been developed for text data, the more interesting aspects of multimedia, namely still images and video, require very different approaches because the data files are several orders of magnitude larger and it is consequently very wasteful of computing resources if one must decrypt the entire file in order to use it.

Computer-based multimedia systems allow the integration of various types of information in a computer. In addition to traditional textual and numeric data, other types of information including graphics (bit-mapped or vector-based), pixel maps, audio- and video clips can be handled in such a multimedia system in digital form. Successful applications include real-time conferencing, group working, multimedia electronic mail, computer-aided distance learning, and interactive video [WILL91]. This paper focuses on security concerns in multimedia, especially authentication, a relatively neglected aspect of multimedia.

Multimedia data can be subverted in several ways. Of interest for us is the fact that transmitted data can be easily copied, distorted, or subverted in other ways by an intruder. For example, a picture sent via satellite it is accessible to anyone who has a TV set and an appropriate antenna [KWYU89] and can be intercepted and subverted during its transmission.

We will address only authentication and verification of multimedia information. While authentication of text is well known, that of image and video information has been ignored in current research area even though image and video are most important in multimedia information. We outline authentication schemes for all three. Sound is not explicitly covered here, but the schemes for still image authentication can be directly applied.

Attacks on the security of a system are best characterized by viewing the function of the system as providing information. There is a flow of information from a source, such as a file or a region of main memory, to a destination, such as another file or a user. This normal flow is depicted in Figure 1a. The other parts of the figure show four general categories of attack [BHAS93]:

- *Interruption*: An asset of the system is destroyed or becomes unavailable or unusable. Examples are destruction of hardware, cutting communication lines, or disabling the file management system.

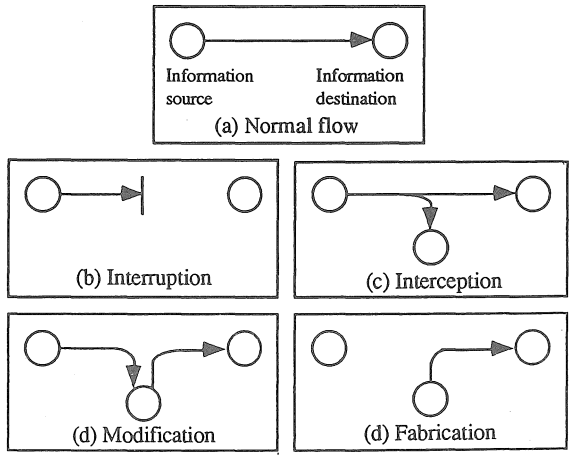


Figure 1. Security Threats

- *Interception*: An unauthorized party gains access to an asset. The unauthorized party could be a person, a program, or a computer. This is an attack on confidentiality. Examples are wiretapping to capture data in a network and illicit copying of files.
- *Modification*: An unauthorized party tampers with an asset. A message or some portion of the message is altered, delayed or recorded, to produce an unauthorized effect. This is an attack on integrity. Examples are changing values in a data file, replacing an image with a fake one in multimedia file, altering a program so that it performs differently, and modifying the content of messages being transmitted in a network.
- *Fabrication*: An unauthorized party inserts complete counterfeit objects into the system. This is an attack on authenticity. Examples are inserting spurious messages in a network or adding records to a file.

We list the major security issues.
Confidentiality: The protection of transmitted data from passive attacks.
Authentication:The assurance that a communication is authentic.
Integrity: The assurance that the information was not modified.
Nonrepudiation: Nonrepudiation prevents either sender or receiver from denying a sent message. Thus, the receiver of a message can prove that it was in fact sent by the alleged sender.

In this paper, we outline methods of how to detect a modification, distortion, or subversion of a multimedia document at the receiver's end. To this end, we present algorithms for generating an authentication code and authentication protocols for text, still image, and video so that the receiver can do the verification and authentication of the received data.

We first give an overview of approaches to text authentication; then authentication algorithms for still image and video are presented. We distinguish the following attacks [SHAF94]:

1. *Disclosure of messages*.
2. *Traffic analysis*:
3. *Masquerade*: Insertion of messages into the network from a fraudulent source.
4. *Content modification*:
5. *Sequence modification*: Any modification in the sequencing of messages.
6. *Timing modification*: Delay or replay of messages.
7. *Repudiation*:

Measures against the first two attacks are in the realm of message confidentiality. Measures against attacks 3 through 6 are generally message authentication. Mechanisms against attack 7 come under the heading of digital signatures. Generally, a digital signature technique will also counter some or all the attacks 3 through 6.

Authentication Functions for Text

Any message authentication mechanism can be viewed as having fundamentally two levels. At the lower level, we have some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as primitive in a higher-level authentication protocol enabling a receiver to verify a message's authenticity.

We outline functions that can be used to produce an authenticator for text message. These functions may be grouped into two classes, as follows:

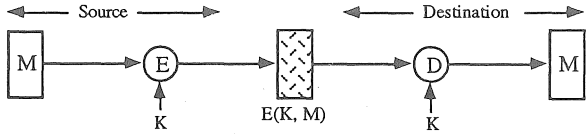
- *Message encryption*: The ciphertext of the entire message serves as its authenticator.
- *Generation function of authentication code:* A public function of the message with a secret key, or a secret function that produces a fixed-length value that serves as the authenticator.

Message encryption by itself can provide a measure of authentication. The analysis differs for conventional and public-key encryption schemes.

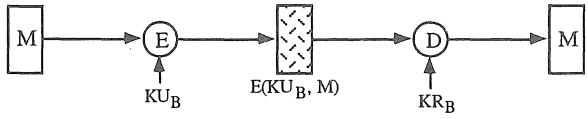
Conventional Encryption

Conventional encryption method, also called symmetric encryption, is the use of the same (or similar) keys for encryption and for decryption: knowing one of the keys enables us to determine the other with little effort. Both keys must be kept secret in conventional encryption. Many conventional encryption methods are available, and one of the best-known is the Data Encryption Standard (DES) proposed by the National Bureau of Standards [LEI82].

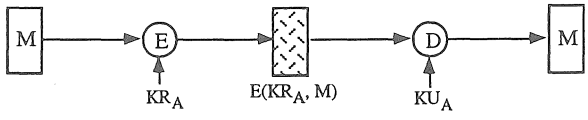
Consider the straightforward use of conventional encryption (Fig. 2a). A message transmitted from source A to destination B is encrypted using a secret key K shared only by sender A and recipient B. If no other party knows K, confidentiality is provided: no other party can recover the plaintext of the message. B is assured that the message was generated by A since A is the only other party that possesses K and thus the only other party with the information necessary to construct ciphertext that can be decrypted with K. Also, if M is recovered, B knows that no. bit of M has been altered, because an opponent that does not know K would not know how to alter bits in a ciphertext to produce desired changes in the plaintext. -- Simple public-key encryption (Fig. 2b) provides confidentiality but no authentication. The other schemes in Figure 2 outline the corresponding variations.



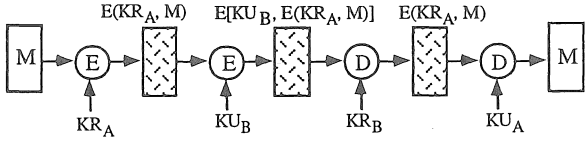
(a) Conventional encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

Figure 2. Basic Uses of Message Encryption

So, conventional encryption provides authentication as well as confidentiality. However, consider exactly what is happening at B. Given a decryption function D and a secret key K, the destination will accept any input X and produce the output $Y = D(K, X)$. If X is the ciphertext of a legitimate message M produced by the corresponding encryption function, then Y is some plaintext message M. Otherwise, Y will be a meaningless sequence of bits. There must be some automated means of determining at B whether Y is legitimate plaintext and therefore must have come from A. The implications of this are profound for authentication. Suppose the message M can be any arbitrary bit pattern. In that case, there is no way for the receiver to determine automatically whether an incoming message is the ciphertext of a legitimate message or not. In general, we require that only a small subset of all bit patterns is a legitimate plaintext. For example, if only one in 10^6 bit pattern is legitimate, the probability that any randomly chosen bit pattern, treated as ciphertext, will produce a legitimate plaintext message is 10^{-6}. For a text file, this is obviously true, because text is inherently redundant. Thus, an automatic tool (based on the probabilities of n-grams in a specific language) can be developed to check automatically, i. e., without human intervention, whether the decrypted message is "meaningful". For more details, we refer to [CHEN95].

Message Authentication Code

An alternative authentication technique uses a secret key to generate a small fixed-size block of data, known as message authentication code (MAC) that is appended to the message. It assumes that two communicating parties, say A and B, share a common secret key K. When A has a message to send to B, it calculates the MAC as a function of the message and the key: C(k, M). The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated authentication code (Figure 3a). If we assume that only the receiver and the sender know the identity of the secret key, and if the received authentication code matches the calculated authentication code, then:

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is

assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
2. The receiver is assumed that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
3. If the message includes a sequence number (as used with X.25, HDLC, TCP, and the ISO transport protocol), the receiver is assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

A MAC generation function is similar to encryption. One difference is that the algorithm need not be reversible, as it must be for decryption. It turns out that because of mathematical properties of the authentication function, it is less vulnerable to being broken than is encryption [DAVI89].

This process provides authentication but not confidentiality, because the message is transmitted in the clear. Confidentiality can be provided by performing message encryption either after (Figure 3b) or before (Figure 3c) the MAC generation algorithm. In both cases, two separate keys are needed, each of which is shared by the sender and the receiver. Typically, it is preferable to tie the authentication directly to the plaintext (Figure 3b).

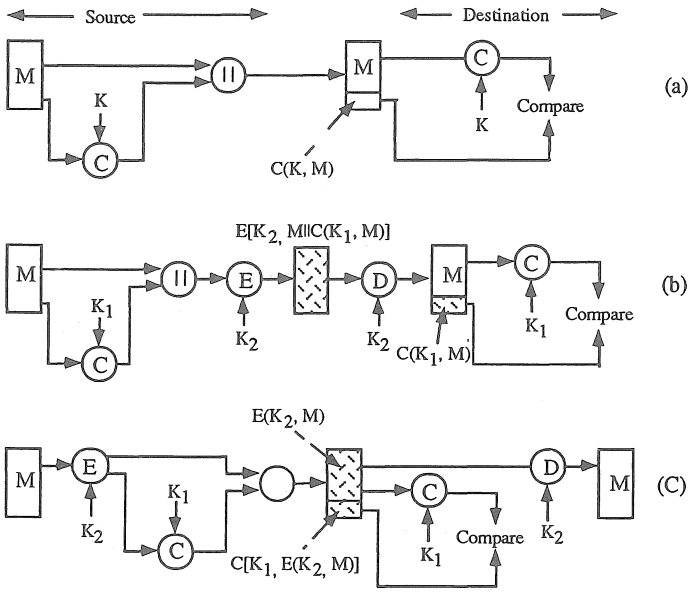


Figure 3. Basic Uses of Message Authentication Function

[DAVI89] suggests three situations in which a message authentication code may be used:
1. The same message is broadcast to a number of destinations, with only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated MAC.
2. One party has a heavy work load and cannot afford the time to decrypt all incoming messages.
3. Authentication of a computer program in plaintext is attractive.

Three other rationales may be added, as follows:
4. Messages need not be kept secret, but they must be authenticated. E. g., the Simple Network Management Protocol Version 2 (SNMPv2) separates the functions of confidentiality and authentication.
5. Separation of authentication and confidentiality functions affords architectural flexibility. One may want to perform authentication at the application level but provide confidentiality at a lower level, e. g., transport layer.
6. One may wish to extend the protection beyond the time of reception and yet allow processing of message contents. With message encryption, protection is lost when the message is decrypted, so the message is protected against fraudulent modifications only in transit but not within the target system.

Requirements for Message Authentication Codes

A message authentication code (MAC) is generated by a function C of the form $MAC = C(K, M)$ where M is a variable-length message, K is a secret key shared only by sender and receiver, and C(K, M) is the fixed-length authenticator. The MAC is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the MAC.

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require 2^{k-1} attempts for a k-bit key. Thus, for a cipher-only attack, the opponent, given ciphertext C, would perform $P_i = D(K_i, C)$ for all possible key values Ki until a Pi was produced that matched the form of acceptable plaintext.

In the case of a authentication code, the considerations are different. The MAC function is many-to-one. The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys. If an n-bit MAC is used, then there are 2^n possible MACs, whereas there are N possible messages with $N >> 2^n$. Furthermore, with a k-bit key, there are 2^k possible keys. Using brute-force methods, how would an opponent attempt to discover a key? If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k > n, i. e., the key size is greater than the MAC size. Then, given a known M1 and MAC1, with $M1 = C(K_1, MAC1)$, the cryptanalyst can perform Mi = C(Ki, MAC1) for all possible key values Ki. At least one key is guaranteed to produce a match of Mi = M1. A number of keys will produce the correct MAC and the opponent has no way of knowing which is the correct key! On average a total of $2^k/2^n = 2^{k-n}$ keys will produce a match. Thus the opponent must iterate the attack. On average, a round will be needed if k = a * n. For example, if an 80-bit key is used and the MAC is 32 bits, the first round will produce about 2^{48} possible keys, the second will narrow the possible keys to about 2^{16} possibilities, and the third should produce only a single key, which must be the one used by the sender. Thus, a brute-force attempt to discover the authentication key is no less effort, and may be more effort, than that required to discover a decryption key of the same length.

Other attacks that do not require the discovery of the key are possible. In assessing the security of a cryptographic MAC function, we must consider what attacks may be mounted against it. Assume that an opponent knows the MAC function C but not K; then the MAC function should have the following properties:

1. If an opponent observes M and C(K, M), it should be computationally infeasible for the opponent to construct a message M' such that C(K, M') = C(K, M).
2. C(K, M) should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that C(K, M) = C(K, M') is 2^{-n}, where n is the number of bits in the MAC.
3. Let M' be equal to some known transformation on M. That is, M' = f(M). For example, f may involve inverting one or more specific bits. In that case, Pr [C(K, M) = C(K, M')] = 2^{-n}.

A number of data authentication algorithms have been proposed. One of the most widely used cryptographic MACs is based on the DES. This algorithm is both a FIPS publication (FIPS PUB 113) and an ANSI standard (X9.17) and appears to meet the requirements specified earlier [SHAF94].

A MAC Based on a Hash Function

A hash value is generated by a function H of the form

$$h = H(M)$$

where M is a variable-length message, and H(M) is the fixed-length hash value. All hash functions operate using the following general principles: the input (message, file, etc.) is viewed as sequence of n-bit blocks and is processed one block at a time to produce an n-bit hash function. The hash value is appended to the message at the source at a time when the message is assumed to be correct. The receiver authenticates that message by recomputing the hash value. Because the hash function itself is not considered to be secret, some means is required to protect the hash value (see Figure 4).

The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. For message authentication, a hash function H must have the following properties (see [NECH92]):

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is easy to compute for any given x, making both hardware and software implementations practical.
4. For any given code m, it is computationally infeasible to find x such that H(x) = m.
5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
6. It is computationally infeasible to find any pair (x,y) such that H(x) = H(y).

A hash function that satisfies the first five properties in the preceding list is refereed to as weak hash function. If the sixth property is also satisfied, then it is refereed to as a strong hash function. Various hash functions for data authentication purpose have been proposed. For example, a hash function based on using a cipher block chaining technique was described in [RABI78]. An algorithm called Snefru was developed by Ralph Merkle for providing good security and ease of implementation on 32-bit processors [MERK90]. However, many algorithms still have weaknesses. Much research [JASP95] is being done in this area.

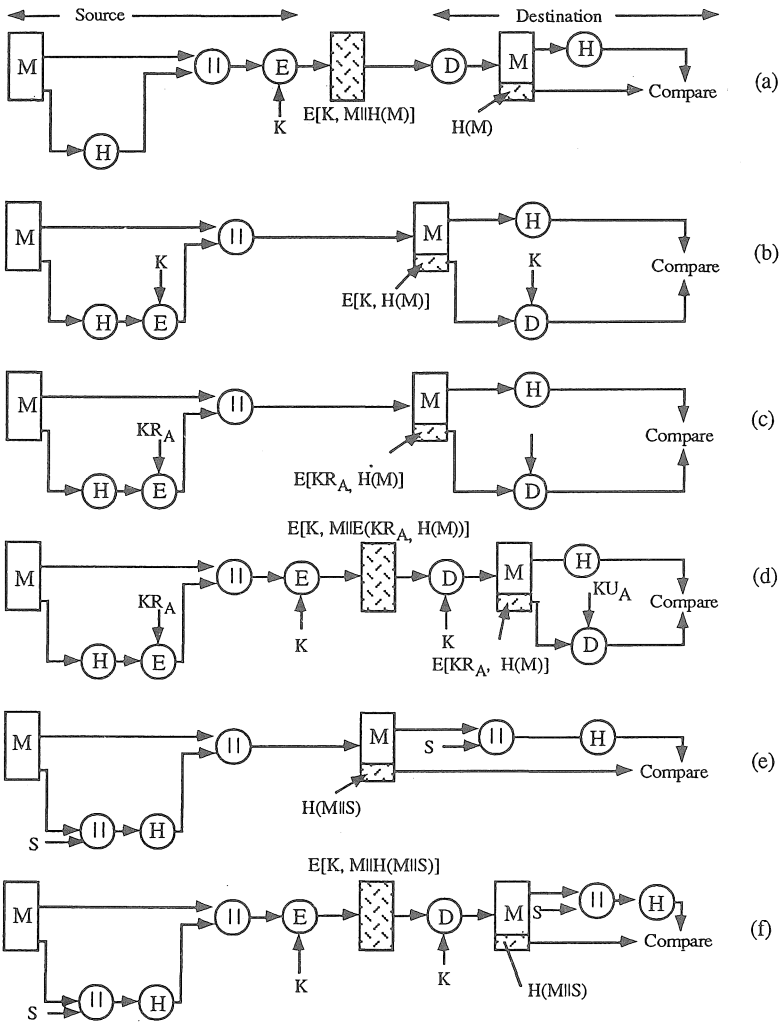


Figure 4. Basic Uses of Hash Functions

Authentication Functions of Image and Video

Usually, image and video files are much larger than text file, so it is impractical to encrypt the entire image and video only for authentication purpose since it would require the sender and the receiver to encrypt and decrypt the message every time a image or a video message is transmitted.

Still Image Authentication

Random Generation

The basic idea of random generation is to choose pixels at random over the whole still image file. The goal is to identify any subversion with a probability approaching 1. We will treat subversions as errors. Thus, a subversion happens when an intruder changes a part (or parts) of a picture such as replacing a head. A subversion usually changes a piece of pixels, but errors could happen in a piece or the errors distribute randomly over the whole image. We assume that the error distribution is even (uniform) on the whole image. We define the error rate r as the average

error per pixel; r = number of errors in the image divided by its total number of pixels. An image is subverted if at least one error is found by checking the witness file. The probability of subversion detection follows the Poisson distribution [JANO88]:

$$P(x;\ \lambda,T)=\frac{(\lambda T)^x\, e^{-\lambda T}}{x!}$$

where T is event internal (number of pixels of the witness file), λ is the arrival rate on average (error rate), x is the number of events (number of errors), and P(x; λ,T) is the probability that x errors are detected. Then, by Poisson's theorem:

$$Pr\ (A\ subversion\ is\ detected) = Pr\ (At\ least\ one\ error\ is\ found) = P(x\geq 1;\ \lambda,T) = 1 - P(0;\ \lambda,T) = 1 - e^{-\lambda T} \quad (1)$$

Suppose a subversion changed 500 pixels in a 640*480=307,200 pixels image. Then $\lambda=500/640*480=0.0016$.

- If the witness file (MAC) is reduced 10,000 times

  $T = 307{,}200 / 10{,}000 \approx 31$ (pixels) and $Pr\ (A\ subversion\ is\ detected) = 1 - e^{-0.0016 * 31} = 0.048$
- If the witness file is reduced 1,000 times

  $T = 307{,}200 / 1{,}000 \approx 308$ (pixels) and $Pr\ (A\ subversion\ is\ detected) = 1 - e^{-0.0016 * 308} = 0.389$
- If the witness file is reduced 100 times

  $T = 307{,}200 / 100 = 3072$ (pixels) and $Pr\ (A\ subversion\ is\ detected) = 1 - e^{-0.0016*3072} = 0.993$

In Formula (1), we see the important relationship between r, T and Pr. We get:

$$e^{-\lambda T} = 1 - Pr \quad or \quad -\lambda T = ln(1-Pr) \quad or \quad \lambda T = -ln(1-Pr) \qquad (2)$$

Thus, given a required probability of subversion detection, its witness file size depends on the error rate. If the error rate is small, the witness file has to be large, if the error rate is large, then the witness file can be much smaller. The witness file size increase with the decrease of λ with the speed of factor ln(1-Pr). Pr is the required probability standard for subversion detection. For instance, if we require the probability of subversion detection needs to be 99.9 percent (acceptable in most cases), then $\lambda T = -ln(1-0.999) = 6.908$. If the error rate r is 0.001, then T = 6.908 / 0.001 = 6908 (pixels). We conclude that the random generation of witness file is a reasonable and acceptable algorithm. Because of (2), we get a higher detection probability and at the same time keep the witness file relatively small.

Transmission Protocol

A message authentication code (witness file) can be used in different ways to provide still image authentication. Suppose A is the sender, B is the receiver. A transmits still image plus authentication code to B. We define:

M:	Original still image file.	M':	Received image file.
C:	Message authentication algorithm (random pixel generator).	C':	Received witness file.
C'':	Generated witness file.	X:	Seed for C.
X':	Received seed.	E:	Encryption algorithm.
R:	Received message.	D:	Decryption algorithm.
K:	Secret key shared by A and B.	KR_A:	A's secret key.
KU_A:	A's public key.	KR_B:	B's secret key.
KU_B:	B's public key.	‖:	Message concatenation

a) *A->B:* $M \parallel E(K, X) \parallel C(X, M)$

When B receives a message M' ‖ R(K,X) ‖ C'(X, M), he does the following:

X' = D(K, R(K,X)); C'' = C(X', M'); if (C'' == C') then "The received image is correct" else "A subversion happened"

In this transmission protocol, only the seed is encrypted using conventional encryption. It is kept secret, so it is computationally infeasible to get M' ≠ M such that C(x, M') = C(X, M) (x is a random seed and x ≠ X). This protocol only provides authentication; no signature is provided. Because K is shared by A and B, B can also "sign" the image.

b) *A->B:* $M \parallel E(KU_B, X) \parallel C(X, M)$

When B receives a message M' ‖ R(KU_B, X) ‖ C'(X, M), he does the following:

X' = D(KR_B, R(KU_B, X)); C'' = C(X', M'); if (C'' == C') then "The received image is correct" else "A subversion happened"

In this protocol, only the (secret) seed is encrypted using public-key encryption. It is computationally infeasible to find M'≠M with C(x, M') = C(X, M) (x is a random seed and x ≠ X). This protocol only provides authentication. Only B can decrypt the seed using his private key. No signature exists because everybody can "sign" a message using B's public key.

c) *A->B:* $M \parallel E(KU_B, E(KR_A, X)) \parallel C(X, M)$

When B receives a message M' ‖ R(KU_B, E'(KR_A, X)) ‖ C'(X, M), he does the following:

X' = D(KR_A, D(KR_B, R(KU_B, E(KR_A, X)))); C'' = C(X', M');

if (C''==C') then "The received image is correct" else "A subversion happened"

In this protocol, the seed is encrypted first using A's private key, then B's public key to provide secrecy.This protocol provides authentication and signature. Only B can decrypt the seed and only A can "sign" the seed using his private key.

d) A->B: M || E(K, X || C(X, M))

When B receives a message M' || R(K,X || C(X, M)), he does the following:

X' || C'(X, M) = D(K, R(K,X || C(X, M))); get X' and C'(X, M); C'' = C(X', M');

if (C'' == C') then "The received image is correct" else "A subversion happened"

The concatenation of seed and witness file is encrypted using conventional encryption. This transmission protocol only provides authentication, no signature is provided. Because K is shared by A and B, B can also "sign" the image.

e) A->B: M || E(KU_B, X ||C(X, M))

When B receives the message M' || R(KU_B, X || C(X, M)), he does the following:

X'|| C'(X, M) = D(KR_B, R(KU_B, X || C(X, M))); get X' and C'(X, M); C'' = C(X', M');

if (C'' == C') then "The received image is correct" else "A subversion happened"

The concatenation of seed and witness file is encrypted using public-key encryption. This protocol only provides authentication. Only B can decrypt the seed and the witness file using his private key. No signature is provided because everybody can "sign" a message using B's public-key.

f) A->B: M || E(KU_B, E(KR_A, X || C(X, M)))

When B receives a message M' || R(KU_B , E(KR_A , X || C(X,M))), he does the following:

X' || C'(X, M) = D(KU_A, D(KR_B, R(KU_B, E(KR_A, X || C(X, M))))); get X' and C'(X, M); C'' = C(X', M');

if (C'' == C') then "The received image is correct" else "A subversion happened"

The concatenation of seed and witness file is encrypted first using A's private key, then B's public-key to provide secrecy. The seed and witness file are kept secret. This transmission protocol provides authentication and signature. Only B can decrypt the seed and the witness file and only A can "sign" a message using his private key.

When a subversion happens in a picture, the block (or blocks) in the subverted part have higher error rates than others, or higher values than the usual transmission error rate. We divide the whole image block by block, all the blocks have approximately the same size. Each block is looked upon as one single unit. We choose pixels randomly in a block. For the details, we refer to [CHEN95]. We have two standards to define that a image was correct and no subversion happened:

- The error rates for all the blocks are close to a known usual transmission error. E. g., if a network's average transmission error rate is 0.01%, we may define as unacceptable error rates in excess of 0.01%.
- Error rates for all blocks are distributed uniformly, which means that all the error rates are almost the same, or in a small, acceptable range.

If an entire picture gets changed, the first standard can detect this subversion, but the second standard may not. In the following, we use the first standard. Define Pr(i) to be the probability that a subversion is detected in block i, β to be the predefined transmission error rate, and τ to be a defined acceptable small changing range. It follows that the more blocks an image is divided in, the more block detection probability terms are added to the subversion detection probability for the image. The more blocks in an image, the smaller the size of each block, so the witness file for each block gets smaller.

Transmission Protocol

Suppose A is the sender, B is the receiver. A wants to transmit the still image plus the authentication code to B. We define:

M:	Original still image file.	M':	Received image file.
C:	Authentication algorithm (random pixel generator).	m:	Numbers of blocks in the image
C'i:	Received witness for block i ($1 \le i \le m$).	C':	Received witness file.
C":	Generated witness file.	Xi:	Seeds of C for each block.
X'i:	Received seeds of C for each block.	E:	Encryption algorithm.
R:	Received message.	D:	Decryption algorithm.
K:	Secret key shared by A and B.	KR_A:	A's secret key.
KU_A:	A's public key.	KR_B:	B's secret key.
KU_B:	B's public key.	||:	Message concatenation
Mi:	Block i in M. $M_1 \| M_2 \| ... \| M_m = M$	X = X1 || X2 || ... || Xm	
X' =	X'1 || X'2 || ... || X'm	C(X,M)=C(X1,M1)||C(X2,M2)||...||C(Xm,Mm)	
C'(X,M) =	C'(X1,M1) || C'(X2, M2) || ... || C'(Xm, Mm)	C"(X,M)=C"(X1,M1)||...||C"(Xm,Mm)	
β:	Transmission error rate.	τ:	A predefined small range value.

Ri: error rate of block i = the number of different pixels between C"i and C'i / the number of pixels in C'i.

a) A->B: M || E(K, X) || C(X, M)

When B receives M' || R(K,X) || C'(X, M), he does the following:

```
X' = D(K, R(K,X)); C'' = C(X', M');
for (i=1;i≤m; i++)
    { Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
      if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
    }
printf("The received image is correct");
```

b) A->B: M || E(KU$_B$, X) || C(X, M)

When B receives M' || R(KU$_B$, X) || C'(X, M), he does the following:

```
X' = D(KR_B, R(KU_B, X)); C'' = C(X', M');
for (i=1;i≤m; i++)
    { Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
      if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
    }
printf("The received image is correct");
```

c) A->B: M || E(KU$_B$, E(KR$_A$, X)) || C(X, M)

When B receives M" || R(KU$_B$, E(KR$_A$, X)) || C'(X, M), he does the following:

```
X' = D(KU_A, D(KR_B, R(KU_B, E(KR_A, X)))); C'' = C(X', M');
for (i=1;i≤m; i++)
    { Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
      if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
    }
printf("The received image is correct");
```

d) A->B: M || E(K, X || C(X, M))

When B receives M' || R(K,X || C(X, M)), he does the following:

```
X' || C'(X, M) = D(K, R(K,X || C(X, M))); get X' and C'(X, M); C'' = C(X', M');
for (i=1;i≤m; i++)
    { Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
      if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
    }
printf("The received image is correct");
```

e) A->B: M || E(KU$_B$, X ||C(X, M))

When B receive the M' || R(KU$_B$, X || C(X, M)), he did the following:

```
X'|| C'(X, M) = D(KR_B, R(KU_B, X || C(X, M))); get X' and C'(X, M); C'' = C(X', M');
for (i=1;i≤m; i++)
    { Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
      if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
    }
printf("The received image is correct");
```

f) A->B: M || E(KU$_B$, E(KR$_A$, X || C(X, M)))

When B receives the M' || R(KU$_B$, E(KR$_A$, X || C(X,M))), he does the following:

```
X' || C'(X, M) = D(KU_A, D(KR_B, R(KU_B, E(KR_A, X || C(X, M))))); get X' and C'(X, M); C'' = C(X', M');
for (i=1;i≤m; i++)
    { Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
      if (abs(Ri - β) > t) { printf("A subversion happened"); exit(1); }
    }
printf("The received image is correct");
```

The strengths and weaknesses of these transmission protocols are the same as the corresponding ones for random generation method, discussed earlier. Protocols a, b, d, and e only provide authentication, Protocols c and f provide a signature as well as authentication.

Video Authentication

Video authentication methods are similar to still image authentication methods, except that the redundancy between successive frames should be considered.

Random Access Frame by Frame

The basic idea is to choose pixels randomly frame by frame. Each frame is looked at as a single unit. Because in video, there is much redundant information between frames, we must not choose the same pixels in a frame; the seeds of different frames need to be different. Suppose a video has S frames, each frame has M*N pixels, and the witness file is T. Define

C: Random number (index) generator.
T_i: The i-th frame ($1 \le i \le S$).
X_i: Seed for frame i ($1 \le i \le S$).
T_{ij}: The j-th pixel in i-th frame ($1 \le i \le S$, $1 \le j \le M*N$).

Then we define

$T_{i,j+1} = C(Xi, Tij) \bmod (M*N)$ $T_i = T_{i1} // T_{i2} // ... // T_{ij} // T_{ij+1} // ...$ $T = T_1 // T_2 // ... // T_S$

where Xi ≠ Xj and |Xi - Xj| ≠ the distance of two adjacent pixels chosen by the random generator C for all 1≤i,j≤S and i≠j.

Probability of Subversion Detection

Assume the error distribution is uniform over the whole video. The error rate r is the average error per pixel,

r = changed number of pixels / total number of pixels in the video.

We say a subversion happened if at least one error is found by checking the witness file. As before we get

$$P(x;\ \lambda,T)=\frac{(\lambda T)^x e^{-\lambda T}}{x!}$$

$Pr(A\ subversion\ is\ detected) = Pr(At\ least\ one\ error\ is\ found) = P(x\geq 1;\ \lambda,T) = 1 - P(0;\ \lambda,T) = 1 - e^{-\lambda T}$

Suppose a subversion happened in a video, a part of the video was changed. Assume that 10 pixels were changed per frame on average. The video lasts 2 hours, there are 30 frames per sec, each frame has 640 * 480 = 307,200 pixels. Then,

$error\ rate\ \lambda = 10 / 640 * 480 = 3.26 * 10^{-5}$

- If the witness file is reduced 10,000 times

  $T = 7200 * 30 * 307{,}200 / 10{,}000 \approx 6.64 * 10^6\ (pixels)$

  $T_i = 7200 * 30 * 640 * 480 / T \approx 10{,}000\ (pixels)\ (1 \leq i \leq S)$

  $Pr(A\ subversion\ is\ detected) = 1 - e^{(-3.26 * 10^{**}(-5) * 6.64 * 10^{**}6)} \approx 1$

- If the witness file is reduced 100,000 times

  $T = 7200 * 30 * 307{,}200 / 1{,}000 \approx 6.64 * 10^5\ (pixels)$ $T_i = 100000\ (pixels)\ (1 \leq i \leq S)$

  $Pr(A\ subversion\ is\ detected) = 1 - e^{(-3.26 * 10^{-5} * 6.64 * 10^6)} = 0.999999999$

- If the witness file is reduced 1,000,000 times

  $T = 6.64 * 10^4\ (pixels)$ $Pr(A\ subversion\ is\ detected) = 1 - e^{(-3.26 * 10^{-5} * 6.64 * 10^4)} \approx 0.8855$

We conclude that the random generation of witness file is a reasonable and acceptable algorithm. We can get a higher detection probability and at the same time keep the witness file relatively small.

Transmission Protocol

The transmission protocol for the video is the same as that for the still image. In the video, each frame has a different seed, in a video 2 hours long with 30 frames per second, there are 2 * 3600 * 30 = 216,000 frames, so we need 216,000 seeds. Transferring all these seeds with encryption will take much bandwidth. Instead, we may use only a single seed (the seed of the first frame); all the other seeds are generated by a mathematical algorithm or another random number generator. We have

$$X_i = C'(X_{i-1}) \bmod (M*N)$$

where X_i is the seed for frame i (1<i≤total frame number), C' is a mathematical algorithm or a random number generator, and M*N is the number of pixels in a frame. In this case, only the seed for the first frame is to be transferred; after the receiver gets the message, he generates other seeds using the same C'.

Random Access with Interframe Compression

In the above random access process, random indices are chosen frame by frame; each frame is looked at as an independent single unit where the seeds for all the frames are different. What happens if we use only a single seed for all frames? It is easy to determine that when one seed is used for all the frames, the positions of selected pixels in a frame are the same as those in all the other frames. For example, if the algorithm is to select pixels with an interval of 100 pixels, the pixels in index positions 1, 101, 201, ..., are picked out in each frame.

There is much redundant information between frames, especially for adjacent frames. Many pixels between adjacent frames are the same. Interframe compression technique always use this feature to eliminate the redundancy, using key frames to store an entire frame, while delta, or difference frames record only interframe changes [WUXY92]. We can develop a single seed random access algorithm with interframe compression to eliminate interframe redundancy: if a pixel selected by a random number (index) generator is the same as the one in the immediately preceding frame, it is ignored; otherwise, it is selected for the witness file.

Suppose a video has S frames, each frame has M*N pixels, X is the seed, C is the random index generator, T is the witness file, P(i,j) is the pixel index position indicating the j-th pixel in the i-th frame. Here is the pseudocode:

```
T = " "; for (i=1; i≤S; i++)
        { P(i,1) = X;
          do if (i==1 || TP(i,j) != Tp(i-1,j)) T = T + TP(i,j);  P(i, j+1) = C(P(i,j)) mod (M*N); j++
          until p(i,j) = X
        }
```

Another variant takes the first frame as key frame, all other frames as delta or difference frames. In the random access process, for a key frame, we select pixels among the entire frame uniformly, but for a delta frame, we only consider the pixels that are different from the immediately preceding delta or key frame. For details, see [CHEN95].

Probability of Subversion Detection

We define the error rate λ as the average error per pixel; then λ = changed number of pixels in the video / total number of pixels in the video; a subversion happened if at least one error is found by checking the witness file.

Interframe motion is a determining factor for the witness file size. The witness file size for a video with a high motion sequence is much larger than for a video with a lower motion sequence. For low motion sequences such as a "talking head" video, greater interframe compression is obtained and the witness file size gets much smaller; thus, for a given witness file size, the generated witness file contains more information about the video than for a high motion sequence, and therefore, the probability of subversion detection gets higher . So, interframe motion is the enemy of interframe compression, it decreases not only the video quality and display rate, but also the probability of subversion detection given a required size of authentication code.-- The transmission protocol is the same as the one of random generation for still images.

Detection of Uniform Distribution

When a subversion happens in a video, some frames in a row or some part in a frame have a higher error rate. The uniform distribution detection algorithm is based on this feature. Divide each considered frame (reference frame) in a video into some roughly equal size blocks. Considering each block as a single unit, we choose pixels at random block by block and store them into a witness file. Take the first reference frame as key frame, all the other reference frames as delta frames. Because of redundancy between frames, when we select blocks in these reference frames, we only need to consider the blocks that are different from the one of the immediate preceding reference frame in the same position.

Suppose a video has S frames, each frame has M * N pixels, each frame is divided into m blocks, one of every s frames is selected as a reference frame. Then, in total we have S/s reference frames and each block has M*N/m pixels. Define C to be the random number (index) generator, X its seed, B(i,j) the jth block in the ith frame, T(i,j) the block in the witness file generated by B(i,j) ($1 \le i \le S$, $1 \le j \le m$), and T the witness file. Here is the witness file generation process in pseudo code:

```
T = " "; for (i=1; i≤S; i = i+s)
        {T = T + "i"; for (j=1; j≤m; j++) if (i==1 || B(i,j) != B(i-s,j)) T = T + "j" + C(X, B(i,j)); }
        }
```

The witness file size is determined by the interval s (among how may frames do we select a reference frame?), the number of pixels in each block, the number of blocks, and the random number generator which decides how many pixels are chosen for the witness file. Again note that high interframe motion is the enemy of the authentication algorithm.

The original video and the seed plus authentication code T are transferred. When the receiver receives this, he generates his own witness file T' with the same seed using the same methods described above. The generated witness file is compared to the received witness file block by block and error rates are calculated for each block (here, we define error rate R_i = different number of pixels between T'i and Ti / number of pixels in Ti for block i, where Ti is a received block in the witness file, T'i is the corresponding generated corresponding block by receiver). The receiver can do this for all the blocks in all reference frames, because all the information randomly accessed is included in the witness file. If needed, the receiver can recover authentication information from a previous part in the witness file. Again, there are two standards to define that no subversion happened:

- The error rates for all the blocks are equal to, or close to a usual transmission error rate.
- Error rates for all blocks are distributed uniformly.

The Probability of Subversion Detection

The probability of a subversion detection is the sum of subversion detection probabilities for all the blocks, which is

$$\Pr(\text{A subversion is detected}) = \sum_{i=1}^{m*S/s} \Pr(Ti) = \sum_{i=1}^{m*S/s} \Pr(|Ri - \beta| > \tau)$$

where β is the predefined usual transmission error rate and τ is an acceptable value. From this formula, we see that the more blocks are got from a video, the more block detection probabilities are added to the subversion detection probability, but this doesn't mean that the subversion detection probability gets larger. The more blocks in a witness file, the more reference frames must be considered or the more blocks must be used in a frame, which means a smaller size for each block. How many blocks and how many pixels in a block should be taken, how many reference frames we need to consider, to get optimized values depends on the real problems. Interframe motion is the enemy for this algorithm; the higher the interframe motion in a video, the larger its witness file size becomes.

Transmission Protocol
These are omitted here; they are analogues of those formulated earlier; see [Chen, 1995].

Conclusion

We have discussed security issues with emphasis on multimedia authentication. Unlike image and video, text itself can be encrypted thereby providing authentication. Encryption using conventional (symmetric) methods such as the DES provides authentication, because only the sender and receiver share a secret key, and so no one else can decrypt and change the message meaningfully. Text encrypted with the public key method can provide not only authentication but also a digital signature.

Even though text itself together with encryption does provide authentication, there are applications in which encrypting the entire text is not appropriate. A text authentication code can be used instead for authentication.There are many algorithms for generating a text authentication code. We reviewed two here: authentication code based on the DES and on a hash function.

In contrast to text, authentication of still images and videos is rarely considered even though images and video are an important part in multimedia. We proposed authentication algorithms for still images and videos. Still images and videos are different from text. They are much larger; usually lossy data compression techniques are applied to them so they can be stored on computer disk and played back effectively. Complete encryption is not an effective way to provide authentication since neither sender nor receiver can afford the time to encrypt and decrypt the whole image or video every time. Effective authentication codes were instead developed here to attach to image or video to provide authentication.

References

[BHAS93] K. N. Bhaskar, *Computer security: threats and countermeasures*, NCC Blackwell, Manchester, 1993.
[CHEN95] F. Chen, Multimedia Authentication, M. S. thesis, Dept. of Computer Science, Univ. of Houston, Dec. 1995.
[DAVI89] D. Davies and W. Price, *Security for computer networks.*, New York: Wiley, 1989.
[JANO88] G. Janos, *Advanced Probability Theory*, M. Dekker, New York, 1988.
[JASP95] B. Jaspan, "GSS-API security for ONC RPC", *Proceedings of the System on Network and Distributed System Security*, San Diego, CA, 1995, 144-51.
[KWYU89] K.W.Yu and T.L.Yu, "Data encryption based upon time reversal transformations," *The Computer Journal*, v. 32, n. 3, 1989, 241-244.
[LEIS82] E. L. Leiss, *Principles of Data Security*, Plenum Press, New York, 1982.
[MERK90] R. Merkle, "A fast software one-way Hash function." *Journal of Cryptology*, v. 3, n. 1, 1990, 43-58.
[NELS92] M. Nelson, *The Data Compression Book*, M&T Publishing, 1992.
[NECH92] J. Nechvatal, "Public key cryptography," in [SIMM92a]
[RABI78] M. Rabin, "Digitalized signatures," in *Foundations of Secure Computation*, DeMillo, R.; Dobkin, D., New York: Academic Press, 1978.
[SHAF94] S. L. Shaffer and A. C. Simon, *Network Security*, AP Professional, Boston, 1994
[SIMM92] G. Simmons, *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.
[WILL91] N. Williams, G. S. Blair and N. Davies, "Distributed multimedia computing: an assessment of the state of the art," *Information Services and Use*, 1991, 265-281.
[WUXF92] X. WU, and Y. Fang, "Lossless interframe compression of medical images", *DCC'92 Data Compression Conf.*, Snowbird, UT, 1992, 249-58.

Estação Multimídia de Atendimento: Uma Abordagem Ergonômica

Marizete Silva Santos
Fernando da Fonseca de Souza
Ana Carolina Salgado

Universidade Federal de Pernambuco, Deptº de Informática
Cidade Universitária - Caixa Postal 7851
CEP 50740-540, Recife - PE - Brasil
E_mail: {mss, fdfd, acs}@di.ufpe.br
Fone: (081)271-8430
Fax number: +55(81)2714925

Resumo

Este trabalho apresenta uma Estação Multimídia de Atendimento que combina vários recursos existentes em estações convencionais com um sensor de presença e um *design* inovador, levando em consideração uma abordagem ergonômica. Este recurso adicional além de facilitar o contato inicial com o computador também melhora a interação homem-máquina, uma vez que o usuário perde a inibição inicial para começar a comunicação.

Uma arquitetura para esses sistemas também é proposta, além da apresentação de testes que validam a aceitação dos usuários com relação a estes sistemas.

Abstract

This work introduces a multimedia service station which combines various resources existing in conventional stations with a presence sensor and an innovative design taking into consideration an ergonomic approach. This additional resource besides of facilitating the initial contact with the computer, also improves the human-machine interaction, once that the user loses the initial inhibition to begin the communication with the machine.

An architecture for these systems is also proposed, and in addition to that validation teste were carried out in order of verifying users acceptance of such systems.

1 Introdução

O homem é um ser em constante evolução. Um exemplo dessa evolução é a descoberta do computador, certamente um dos maiores avanços da humanidade na área tecnológica. Dado esse avanço podemos hoje acoplar diversos recursos de mídia em uma única estação multimídia para atendimento ao público. Na atualidade existem estações multimídia desenvolvidas para as mais diversas aplicações, temos por exemplo, terminais de atendimento ao público que prestam serviços como: informações turísticas em hotéis, supermercados, farmácias, reservas aéreas e outros.

Apesar desse progresso onde os terminais de atendimento ao público possuem os mais diversificados recursos, como tela sensível ao toque, CD-ROM, placa de áudio, alto falantes, entre outros, muitas estaçõs existentes hoje ainda carecem de uma melhor interatividade com o usuário, um *designer* que deixe o usuário neófito mais desinibido e também de uma melhor interface para comunicação com o computador. Os aplicativos multimídia existentes hoje, parecem ainda não explorar todas as potencialidades que as novas tecnologias oferecem de forma integrada e atraente, principalmente no campo dos terminais de atendimento ao público.

O objetivo deste trabalho é apresentar uma Estação Multimídia de Atendimento - EMA, um sistema computadorizado concebido com base em considerações ergonômicas que acopla vários recursos de mídia em um único terminal de atendimento ao público, incluindo um sensor de presença, recurso inovador, às estações multimídia.

Inicialmente é apresentado uma abordagem ergonômica; em seguida uma seção sobre a EMA com seus recursos; na seção seguinte são mostrados os testes realizados com vários usuários comprovando que o novo recurso (sensor) melhora a interação homem-máquina, uma vez que o usuário perde a inibição inicial ao começar a comunicação. Por fim, é apresentada a conclusão geral do trabalho.

2 Abordagem Ergonômica

A adequada adaptação do usuário a algum tipo de sistema depende fundamentalmente da concepção de software. Com uma concepção deficiente, o usuário pode não conseguir adaptar-se ao sistema, acarretando com isso altas taxas de erros, desinteresse e frustação, gerando consequentemente prejuízos. Objetivando construir *software* eficiente, vários aspectos da sua

concepção têm sido abordados na Engenharia de Software [1,2,3]. Atualmente existem duas abordagens para a concepção do *software*: tradicional e ergonômica.

O método ergonômico de desenvolvimento de sistemas prioriza o conhecimento do usuário e da tarefa como elementos básicos, enquanto que o tradicional se apoia no sistema de informação e estrutura organizacional, considerando apenas superficialmente os aspectos humanos envolvidos [4].

Dado a inviabilidade de construção de sistemas que se adequem simultaneamente a todos os usuários, é de fundamental importância o conhecimento das necessidades do usuário e conhecimento da tarefa a ser executada [5]. O conhecimento do usuário diz respeito à experiência anterior e habilidades do usuário, levando-se em consideração que o sistema deve ser flexível de modo a se moldar às evoluções de experiências adquiridas pelo usuário. No ciclo ergonômico, a primeira etapa definida é o estudo preliminar do público alvo a que se destina. Definido o público alvo, é feito um levantamento preliminar para obter informações que sejam relevantes ao trabalho. Este sistema foi totalmente desenvolvido levando em consideração o ciclo ergonômico de concepção de *software*.

Outro aspecto de suma importância é a interface[1], meio através do qual se efetiva a comunicação entre o ser humano e o computador. Tem-se comprovado que muitos dos problemas com os sistemas de computação na atualidade são em função de interfaces mal projetadas [6]. Sistemas que apresentam uma boa interface são economicamente mais viáveis e mais aceitos comercialmente, pois reduzem o tempo de treinamento necessário, diminuindo assim, os custos da empresa, os quais já envolvem custos de software e hardware [7].

No *design* construído para a EMA, durante todo o desenvolvimento do projeto tentou-se levar em consideração os aspectos estéticos e funcionais do produto e algumas recomendações ergonômicas [8]. Dentre as várias definições de ergonomia, optamos neste trabalho pela definição proposta por Ilida [9] que cita como objetivos práticos da ergonomia a segurança, o bem-estar e o conforto do usuário. O conforto proposto vai desde o dimensionamento do produto até o desenvolvimento do *software* do sistema. Vários aspectos são analisados visando atingir este objetivo, com por exemplo, a inclinação ideal para o monitor de vídeo deve estar de acordo com o ângulo de visão do usuário e disposição das informações na tela. Neste trabalho deu-se uma ênfase

[1] Interface ser humano-computador é aqui referida simplesmente como interface

em ergonomia justamente porque se pretendia chegar a um *design* que fosse confortável, seguro, amigável e o mais natural possível para o usuário.

Para se conseguir este bem-estar do usuário se faz necessário lidar com a antropometria, que trata das medidas físicas do corpo humano. Ela está dividida em estática, dinâmica e funcional. No *design* elaborado a opção foi feita pela antropometria estática, que é aplicada ao projeto de objetos sem partes móveis ou com pouca mobilidade.

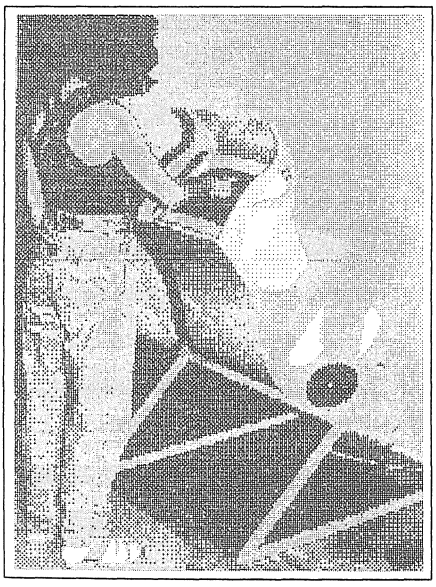
A nível estético, foi feito um estudo de vários *designs* existentes no mercado. Não foi imposta uma preferência formal, mas adotou-se uma forma que era a mais agradável possível visando atrair o usuário.

3 Estação Multimídia de Atendimento

Nesta seção é apresentada a Estação Multimídia de Atendimento, destacando-se a arquitetura do sistema e o sensor de presença.

3.1 Estações Multimídia

Estação multimídia de atendimento é um sistema computadorizado que trata com vários meios, tais como, texto, gráfico, som e imagens. Existem diversas estações multimídia espalhadas pelo mundo para as mais diversas aplicações [10, 11], mas não existem no entanto, até o momento, estações multimídia com algum recurso para detectar a presença do usuário e que também tenha um *design* ergonômico que possa ser acessado por deficientes físicos (paraplégicos).



A Estação Multimídia de Atendimento - EMA (Fig. 1 ao lado), desenvolvida neste trabalho, é um sistema computadorizado que acopla diversos recursos de mídia em um único terminal para atendimento ao público em vários tipos de serviços. Como a comunicação entre o homem e o computador vem sendo feita na maioria das vezes num estilo pouco atraente e

inacessível para usuários pouco experientes, é proposto aqui um novo recurso - **um sensor de presença** - uma interface alternativa que tem como objetivo detectar a presença do usuário antes que este tenha que utilizar algum procedimento inicial para interagir com a máquina. Combinado a essas características, é proposto ainda um *design* ergonômico e inovador que acomoda todos os componentes físicos do sistema [12].

3.2 Arquitetura da EMA

A primeira motivação para estabelecer a interação entre o sistema e o homem é feita através do ***design*** da estação, que tem como objetivo suportar os equipamentos e servir de estímulo inicial para aproximação do usuário.

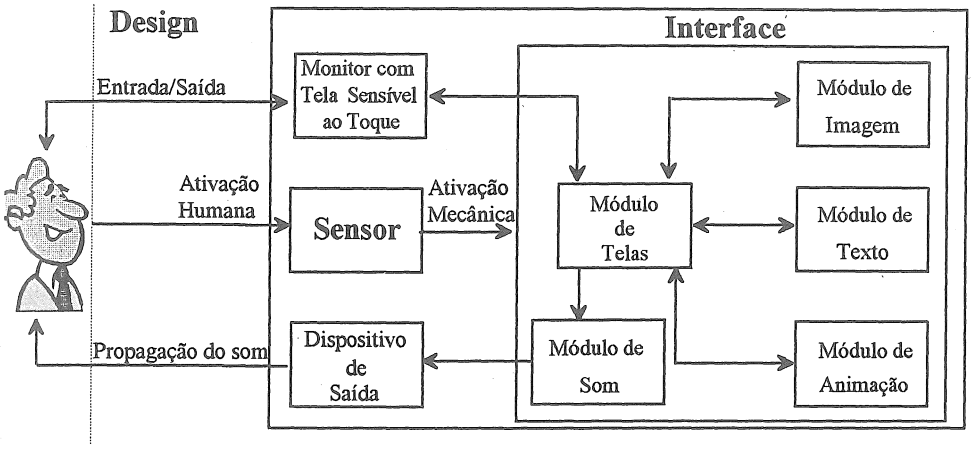


Figura 2: Arquitetura da EMA

Ao se aproximar da EMA, o **sensor** [13] detecta a chegada do usuário e ativa os módulos da interface. A interação com o sistema pode ser feita através dos vários estilos de interface. Neste trabalho os estilos de interface utilizados foram a manipulação direta, menus e ícones.

O **módulo de telas** evoca os demais módulos (ver Fig. 3). Ao ser solicitado através de uma ação do usuário pelo monitor de tela sensível ao toque, este pode chamar todos os módulos em conjunto ou parte deles. Neste módulo são definidas todas as telas do sistema. Estas telas foram construídas levaram em consideração alguns princípios de desenho assim como também critérios sobre a utilização de cores visando obter algumas qualidades ergonômicas requeridas

como a simplicidade para aprender e utilizar, facilidade de memorização, rapidez e confiabilidade. Alguns princípios como equilíbrio visual, anomalia, simetria e o posicionamento do título centralizado para enfatizar a organização lógica da tela são mostrados na figura 3.

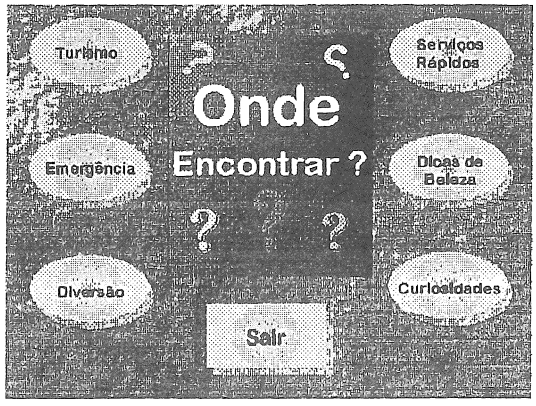


Figura 3: Exemplo de uma tela do sistema

O **módulo de som** ao receber uma chamada do módulo de telas, envia o som que será propagado através do dispositivo de saída. Neste módulo foram desenvolvidos estudos sobre a emissão de arquivos de som, o que propicia uma comunicação ainda mais amigável entre o usuário e a máquina. Os arquivos de som tratados aqui são do tipo WAVE e MIDI [14].

O **módulo de animação** é responsável pelas animações que serão enviadas ao módulo de telas a serem exibidas no monitor com a tela sensível ao toque para interação com o usuário.

O **módulo de texto** é responsável por mostrar as informações textuais no monitor, solicitadas pelo módulo de telas.

O **módulo de imagem** ao ser evocado pelo módulo de telas retorna as imagens que serão visualizadas pelo usuário através do monitor com uma tela sensível ao toque. Esse recurso proporcionará ao usuário um "teclado inteligente" que pode ser reconfigurado a cada instante em que for necessário, durante a comunicação. O principal objetivo é prover uma interface que seja a mais agradável possível, eliminando o uso de várias teclas desnecessárias. Além disso, haverá uma maior comodidade por parte do usuário, o qual é poupado da digitação de nomes ou números.

3.3 Sensor de Presença

O sensor de presença é um dispositivo de entrada de dados que tem como função principal detectar imediatamente a presença do usuário e acionar o programa no monitor de vídeo.

Tão logo a comunicação é estabelecida, começa-se a interação com o computador. A utilização do sensor de presença na estação torna a comunicação mais simples e natural uma vez que o usuário se desinibe, facilitando assim a interação com a máquina.

Apesar de existirem vários tipos de sensores que podem ser acoplados às estações, neste trabalho implementamos um sensor de pressão por apresentar baixo custo e simplicidade de construção.

O funcionamento do Sensor de presença está representado no diagrama de blocos da figura 4. A chave normalmente está desativada. Quando o usuário se aproxima da estação, seu peso a aciona gerando um sinal que será tratado pelo módulo seguinte. Este bloco funciona como uma espécie de filtro, que ao receber o sinal faz o tratamento deste, evitando que ruídos externos provoquem picos e o sistema seja ativado equivocadamente. O sinal ao sair deste bloco será interpretado pelo bloco de comunicação, que tratará esse a um nível adequado para ser trabalhado pelo computador. A nível elétrico, o sensor de presença está representado na figura 5 abaixo.

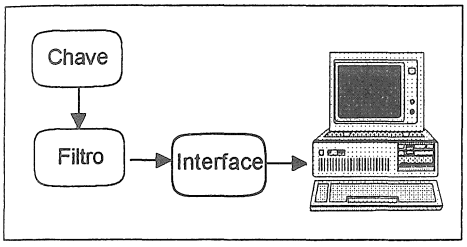


Figura 4:Diagrama de bloco do sensor

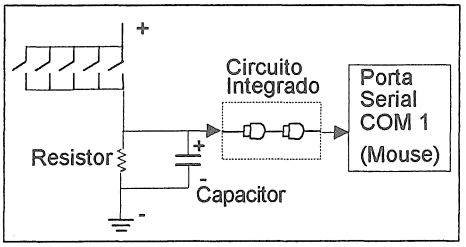


Figura 5:Diagrama elétrico do sensor de presença

4 Testes de Validação

Aqui serão apresentados os resultados juntamente com uma análise das pesquisas realizadas com 102 pessoas durante três apresentações públicas, objetivando tanto avaliar o nível de aceitação do sistema como também buscar sugestões para melhoria na comunicação inicial com a máquina. Todas as apresentações foram realizadas na cidade do Recife, Estado de Pernambuco.

O método de análise empregado foi o mesmo para todas as pesquisas. Segundo Anamaria [15] as pesquisas podem ser divididas em 2 grandes grupos: um de pesquisa descritiva e outro de pesquisa experimental. A pesquisa experimental utiliza métodos experimentais enquanto que a pesquisa descritiva usa técnicas padronizadas de coletas de dados, como o questionário, a entrevista e as observações.

Um dos objetivos desta pesquisa é justamente detectar a opinião da população sobre o sistema proposto, visando uma melhor interação do homem com o computador. Neste sentido, adotou-se aqui a pesquisa descritiva, habitualmente aplicada quando se tem por objetivo levantar opiniões, atitudes e crenças de uma população.

Entre os vários tipos de entrevistas existentes, deu-se preferência à entrevista semi-estruturada, aplicada a partir de um pequeno número de questões abertas. Esta foi adotada levando em consideração o caráter das apresentações públicas onde existia um grande número de pessoas transitando constantemente para observar os vários trabalhos expostos, uma após outra seguidamente .

4.1 Apresentações realizadas

1) A primeira apresentação da estação foi feita durante a IX Escola de Computação, realizada na UFPE, em julho/94. Nesta apresentação o usuário pisava diretamente sobre o sensor. O objetivo desta, foi verificar qual o nível de aceitação do trabalho por parte de usuários especializados.

Antes que o usuário se aproximasse da EMA, na tela do computador era mantida uma mensagem animada convidando-o a se aproximar. Quando este se aproximava e pisava sobre o sensor uma música era tocada e imagens gráficas apareciam. Ao final da apresentação surgia uma tela de agradecimento pela participação. O público alvo desta demonstração foi a classe universitária de um modo geral (professores, alunos, funcionários administrativos) e alguns empresários.

2) A segunda apresentação da estação ocorreu em setembro do mesmo ano durante a inauguração de um museu interativo de ciência[2]. O público alvo desta vez foi mais diversificado: alunos e professores de 1º, 2º e 3º graus, políticos e pessoas da comunidade. Percebeu-se que tanto as crianças quanto os adultos usaram o computador diversas vezes sem apresentar nenhum constrangimento.

A implementação dessa apresentação consistiu da descrição da proposta geral do trabalho, isto é, havia um som de fundo e a pessoa ao se aproximar da EMA pisava sobre o sensor que não estava visível ao usuário. Ao pisar sobre o mesmo, ouvia-se uma voz feminina falando

[2] O museu interativo de Ciência é um projeto desenvolvido pela Secretaria de Ciência e Tecnologia do Governo do Estado de Pernambuco que tem como meta oferecer às escolas de 1º e 2º um museu tecnológico onde os alunos e professores possam utilizar muitos recursos que não estão disponíveis em suas escolas.

sobre a tela sensível ao toque, o *design*, o próprio sensor e o som. Ao mesmo tempo em que ouvia-se a voz, iam surgindo imagens gráficas de cada componente descrito. Assim, como na primeira apresentação, havia um recurso auditivo e gráfico utilizado na interface para chamar a atenção dos usuários. Notou-se aqui também uma boa aceitação por parte do usuário ao utilizar o sistema, não se intimidando perante o computador.

3) A terceira apresentação da estação se deu em outubro/94 no 3º Congresso de Informática e Telecomunicações do Nordeste e na 3ª Feira de Informática do Nordeste. Nesta apresentação o procedimento para ativação do sensor foi o mesmo utilizado na apresentação anterior. Quanto a implementação, houve apenas algumas alterações de algumas telas gráficas e animação. Aqui se observou também uma boa aceitação por parte do usuário. O público alvo deste teste foi os participantes do Congresso e da Feira, que era composto de usuários leigos e especializados, pesquisadores e empresários da área de Computação. Nesta apresentação o sensor não estava visível ao usuário.

4) Diante dos resultados positivos obtidos durante as três primeiras apresentações e uma vez que a resposta do usuário é de fundamental importância para este trabalho, uma nova apresentação do trabalho foi feita durante a III Jornada de Informática, em junho/95 na UFPE, tendo como objetivo principal buscar melhorias quanto a navegação, utilização das cores e aceitabilidade da estação junto a usuários especialistas.

A aplicação utilizada nesta implementação foi direcionada a um Shopping Center. Para a elaboração da implementação foi feita uma entrevista inicial com alguns usuários visando detectar as reais necessidades desses, fator de fundamental importância para o ciclo ergonômico de desenvolvimento de software [5].

Nesta implementação havia um som de fundo e a pessoa ao se aproximar da EMA pisava sobre o sensor, visível ao usuário. Ao pisar sobre o mesmo, ouvia-se uma voz feminina explicando como utilizar o sistema. Em seguinda, a tela principal era apresentada e a voz perguntava ao usuário o que ele necessitava. O mesmo fazia sua escolha entre as opções: filmes em cartaz, moda (criança e adulto), sapatos e músicas da atualidade. À medida que o usuário selecionava as opções, novas telas iam surgindo gradativamente até o último nível da árvore.

4.2 Análise dos Resultados da Pesquisa

Os resultados obtidos a partir da aplicação dos questionários, respondidos por alguns usuários são apresentados a seguir. Por se tratar de questões abertas, as respostas dos usuários foram reunidas em determinados grupos.

Com relação à primeira questão que pergunta ao usuário se este encontrou alguma dificuldade para interagir com a Estação, 100% dos entrevistados afirmaram não terem encontrado nenhuma dificuldade inicial para interagir com a Estação.

A segunda questão oferece ao entrevistado as opções em classificar o sistema em fácil, difícil ou regular; obteve-se um percentual de 100% afirmando que o sistema é fácil de ser usado.

A terceira pergunta compara a EMA com outras estações multimídia abordando as vantagens e/ou desvantagens oferecidas. A resposta dos usuários pode ser vista na figura 6.

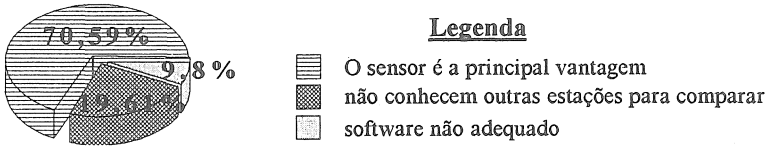


Figura 6: Comparação da EMA com outras estações

Na quarta questão objetiva-se descobrir se o usuário percebeu a presença do sensor sob os pés, e em caso afirmativo, qual foi a sua reação. 50% dos entrevistados perceberam a presença do sensor. A reação dos usuários que perceberam a presença do sensor está discriminada no gráfico da figura 7.

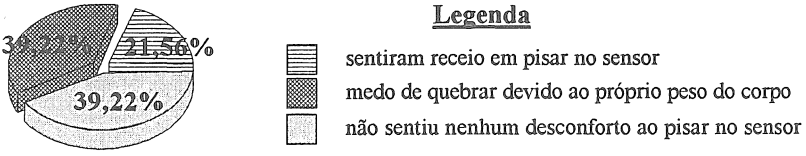


Figura 7: Reação dos usuários que perceberam o sensor de presença

A resposta à quinta questão que solicita sugestões de melhoria para o sensor pode ser vista na figura 8 da página seguinte.

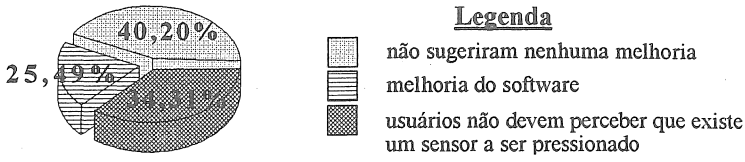


Figura 8: Sugestões para melhoria da interface

Embora não houvesse condições de entrevistar a maioria dos participantes, em paralelo com estas entrevistas foram feitas observações diretas no local das apresentações. Um fato que chamou a atenção foi a atitude demonstrada por determinados participantes de testar o sensor mais de uma vez e convidarem outras pessoas a experimentarem também. Este fato também leva a acreditar que os usuários alegravam-se com a interface, proporcionando consequentemente satisfação aos mesmos.

As implementações feitas durante o decorrer do trabalho utilizaram o HSC Interactive, uma linguagem de autoria indicada para ser utilizada em qualquer aplicação multimídia interativa que combina gráficos, animação e áudio. Ela usa ícones para representar um fluxograma ou a estrutura de uma aplicação. O processo de desenvolvimento da aplicação se completa quando o conteúdo é associado a cada ícone na estrutura através de uma série de quadros de diálogo.

A plataforma de hardware disponível para as implementações da EMA foi baseada em microcomputador com o padrão IBM-PC, contando com Sistema Operacional MS-DOS e ambiente de programação Windows 3.0 da Microsoft. Para a construção das telas foram utilizados CorelDraw, Ami Pro e algumas ferramentas do ambiente Windows, decisivas para o êxito do trabalho.

5 Conclusões

Este trabalho apresentou a proposta de uma estação multimídia de atendimento, um sistema computadorizado que acopla vários recursos de mídia em um único terminal de atendimento ao público, dando ênfase em um sensor para detectar a presença do usuário e um design ergonômico que leva em consideração os deficientes físicos (paraplégicos), recursos esses de extrema importância para a melhoria da interação ser humano-computador.

Com a realização da pesquisa foi possível comprovar que a comunicação inicial tem uma importância fundamental no processo de comunicação com a máquina. Este contato inicial é o principal obstáculo a ser vencido no uso de estações multimídia de atendimento ao público.

A pesquisa também revelou que o público alvo fica menos inibido quando não tem conhecimento do mecanismo de ativação, ou seja, o design da EMA deve "embutir" o sensor de forma que este não seja percebido pelo usuário.

Notou-se também que o uso de um recurso, como o sensor de presença, em estações multimídia visando facilitar o primeiro passo na interação ser humano-computador, de modo a livrar o usuário de ter que utilizar algum procedimento inicial, é de extrema importância também para melhorar essa interação.

A avaliação feita a cada apresentação do protótipo da estação reforçou a tese de que um programa construído de acordo com as necessides dos usuários atinge de forma mais eficiente os objetivos do software. Esse cuidado tem seu último esforço no software final de implementação, não implicando, entretanto, que foram exauridas todas as possibilidades de melhoria.

6 Referência Bibliográfica

[1] Sommerville, I. Software Engineering. New York: Addison-Wesley, 1992.

[2] Eason, K. Ergonomic Perspectives on Advances in Human-Computer Interaction. *Ergonomics*, 34(6), 721-741, 1991.

[3] Iochpe, C. e Figueiredo, E. M. P. Estudo das Interações Humanas no Processo de Desenvolvimento de Software. Simpósio Brasileiro de Engenharia de Software. Recife, outubro 1995, 327-341.

[4] Righer, Carlos Antonio Ramirez. Aplicação de Recomendações Ergonômicas ao Componente de Apresentação da Interface de Softwares Interativos. UFSC. Programa de Pós-Graduação em Engenharia de Produção da UFSC, dissertação de mestrado, 1993.

[5] Johnson, H. and Johnson, P. Integrating Task Analysis into System Design: Surveying Designers' Needs. *Ergonomics*, 32(11), 1451-1467, 1989.

[6] Brad, A. Myers. Challenges of HCI Design and Implementation. Interactions, 1(1), 73-83, 1994.

[7] Scapin, D.L.; Pascale, R. e Agnes Pollier. La Conception Ergonomique d'interfaces: problemes de methode. INRIA/CNET, 1988.

[8] Cusham, W. H. Human Factores in Product Design. New York: Elsévier, 1991.

[9] Ilida, Itiro. Ergonomia: Projeto e Produção. São Paulo: Edgard Blucher, 1990.

[10] Inovações para Automação Comercial. *Telebrasil*, 66, Novembro/Dezembro de 1993.

[11] Filho, S. H. Projeto de Informações Turísticas em Multimídia. INFOBAHIA/93. *IEEE Brasil Conf' 93*. Salvador, 374-379, 1993.

[12] Santos, Marizete S.; Sousa, Fernando F. e Salgado, Ana C. An Alternative Interface for Multimedia Service Stations. Human-Computer Interaction Interact'95. New York: Chapman & Hall, 237-240, 1995.

[13] Santos, Marizete S. Estações Multimídia: Projeto e Inclusão de Novos Recursos. Dissertação de Mestrado, Universidade Federal de Pernambuco, Departamento de Informática, 1995.

[14] Wodaski, Ron. Multimídia. Ciência Moderna: Rio de Janeiro, 1993.

[15] Moraes, Anamaria de. Pesquisa em Design: Tecnologia e Métodos Científicos. *Anais do P&D Design 94*, Rio de Janeiro, ano II, IV, 3-14, 1994.

Testing the usability of an Hypermedia Encyclopedia on Data Networks

Lic. Javier Díaz, jdiaz@unlp.edu.ar
Lic. Ivana Harari, iharari@ada.info.unlp.edu.ar
Lic. Claudia Queiruga, claudiaq@ada.info.unlp.edu.ar
LINTI. Dto. de Informática. UNLP. Calle 50 y 115 s/n 3° piso La Plata. Argentina.
Tel/Fax: 54-21 42730/ 54-21 257240, also supported by CIC - BA

Abstract

The usage of hypermedia systems has several side effects mainly regarding human factors. Enhancing the power of expressiveness and multiple user interface's options are their most outstanding features, but the gap between the intended benefits and their real effect is hardly considered. Comparison of basic tools for exploring and searching information in hypertexts has been elaborated in several publications [Frei92], [Guinan92]. Nonetheless, the introduction of additional tools to simplify the understanding and usage of hypertexts had not been explored at all. New tools were carefully examined when creating the **MHEF** model [Díaz95], that is an Hipertextual Model with Statistics and Filters, defined in our Laboratory (LINTI at the University of La Plata) and implementing an application in 1995. This model includes tools for information retrieval and for the customizing and personalizing of the system. So, the purpose of this paper is to demonstrate the effect of these additional tools for hypertexts with regard to the user's perspective in order to improve the human-computer interaction and to denote all their potential features. This is achieved through an evaluation process carried out with different kinds of readers. To perform the testing we have chosen an application called Hypermedia Encyclopedia on Data Networks [Díaz94a] [Díaz94b]. We define a test for usability evaluation over a range of thirty participants [Petre95] [Kara95].

The conclusions reveal the real advantages of using additional tools that facilitate the user interaction and present hints to be considered in future versions of **MHEF**.

Descriptive synthesis of MHEF

MHEF (Hypertextual Model with Statistics and Filters) incorporates a set of tools constructed in order to help readers in the information retrieval process [Frei92] [Guinan92] [Belkin92] [Inchimura93] and to allow for customizing and personalizing the system.

The tools for data retrieval incorporated in MHEF are: **Recommended Lecture paths by the hypertext's author**, **Dynamic guided tours** that are related with the popularity of the hypertext's nodes, **Key-word filters** that allow direct access to specific nodes of the system [Díaz95].

The **Recommended Lecture path tool** allows users to take a glance at the basic subjects involved in the hypertext. It is determined by the author of the hypertext. In the application under consideration (the Encyclopedia on Data Networks), this tool brings a brief trace to the most relevant information about data networks. It constructs a static linear representation of the hypertext graph, including the creation of new appropriate links to connect the most important nodes. This tool facilitates the comprehension of the ideas included in the hypertext.

The **Dynamic guided tour tool** shows the nodes and links most visited by all the readers of the system, in a hierarchical structure [Rivlin94] [Trigg88] [Marshall89]. The links shown are those that relate the most popular nodes and also, new ones are generated if it is necessary for the connectivity of the overall tree structure. This a dynamic and evolutive mechanism which depends on the empirical usage of the system. It was incorporated in **MHEF** for helping users to visualize in a simply manner the most advised information of the hypertext (remarking the most popular concepts referenced by the universe of readers).

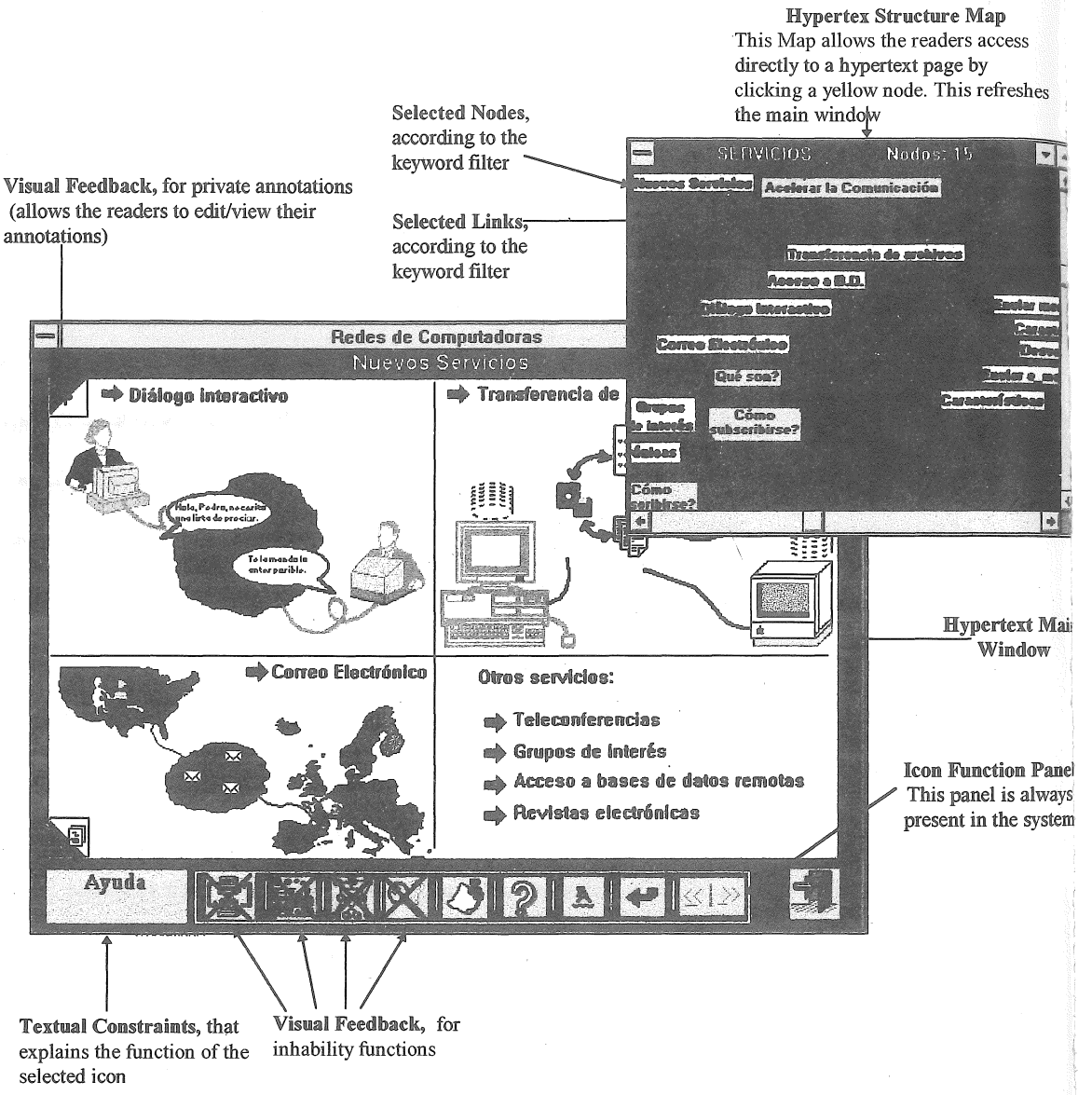
The **Key-word filters tool** allows the recovering of nodes related to a specific topic selected by the user allowing the direct access to nodes according to their contents. It constructs a subgraph with the nodes classified by the key-word filter selected and the active links are only those that connect them. The fundamental purpose of this tool is to simplify the information retrieval for end users.

When one of these tools is selected by a reader, a simplified domain of the hypertext is activated as the result of the tool applied. Users can navigate in the resulting domain structure as usual or they can work with a map constructed in order to bring a visual representation of that structure and to permit the direct access to the hypertext pages.

The other tools let the users set up individual views of the hypertext's domain. They include **Filters according to the user's profile** [Sokolnicki88], and the possibility to have **Private (personal) annotation** in the hypertext's nodes [Díaz95]. The first one consists on a classification of the potential users of the system in well-defined categories partitioning the hypertext domain according to them. So, the system information is organized in different hypertext graphs concerned to each kind of readers. The last one, the **Private annotation tool** allows users to incorporate private notes into the node's contents like illustrations, figures, comments, phrases. For doing this, a simple text editor and a graphic library are provided. With this tool, hypertext readers can enlarge the semantic (adding a personal flavor) of the nodes.

In order to evaluate the usability and adequacy of these tools and their influence in the global system, **MHEF** model was applied to a domain of data networks, creating an HYPERMEDIA ENCYCLOPEDIA ON DATA NETWORKS (a copy is available upon demand at the LINTI).

The look and feel of the added tools was designed carefully using the most of the visual interaction paradigms like direct manipulation, visual maps, colors, constraints, icons, gestures, etch [Petre95]. The next figure displays part of the visualization of the **Key-word filter tool.**



Usability test

The usability-evaluation was performed over a range of thirty participants classified as novice and experienced users [Petre95] [Kara95]. They were compelled to solve five goals that constitute different types of search in the hypertext domain, analyzing the usability of the data retrieval tools.

The problems were:

Goal-1: *to search information related to an specific point,*

Goal-2: *to synthesize a topic,*

Goal-3: *to explore some theme exhaustively,*

Goal-4: *to inquire the most popular subjects consulted in the hypertext* and,

Goal-5: *to examine the global subject of the hypertext.*

Below, we present the Reference Table with the most appropriate (M-A), acceptable (A), not always admissible (N-A-A) (relying on the usage of the system), and unacceptable (U) tools determined per objective.

Objective	Navigation Tool	Guided Tour Tool	Key-word Filter Tool	Rec. Lecture Path Tool
Goal-1:	(A)	(N-A-A)	**(M-A)**	(N-A-A)
Goal-2:	**(M-A)**	(N-A-A)	(A)	(N-A-A)
Goal-3:	(A)	(N-A-A)	**(M-A)**	(N-A-A)
Goal-4:	(U)	**(M-A)**	(U)	(U)
Goal-5:	(A)	(N-A-A)	(U)	**(M-A)**

The responses obtained from the users were matched with this table and the results are listed below:

a) Results about Novice readers

For the goals *To search information related to an specific point* (Goal-1) and *To explore some theme exhaustively* (Goal-3) where the best choice was the *Key-word filters tool* only a 15 % and a 0 % accordingly of the novice readers used it. Overall, they choose **Navigation tool** (close of a 60 %) and **Recommended Lecture Path** (close of a 25 %).

For the goal *To synthesize a topic* (Goal-2) where the best choice was **Navigation tool**, a 60% used this appropriate tool. The rest of the readers selected the **Recommended Lecture Path tool** (33 %) and **Key-word filters tool** (7%) to perform this task.

For the goal *To inquire the most popular subjects consulted in the hypertext* (Goal-4) where the best option was **Guided Tour tool**, only a 20 % of the readers used it and about a 80 % of them can't perform the task.

For the goal *To examine the global subject of the hypertext* where the best tool was the **Recommended Lecture Path tool** a 60 % of the novice users selected it. The rest of the readers used **Navigation tool** (about 33 %) and **Key-word filters tool** (7%).

We conclude that the novice readers always use navigation and recommended lecture path tools for every given goals. They don't use the additional tools provided for the system that allow direct access to the hypertext information. They prefer to perform the task in an exhaustive way selecting the navigation tool or in a full guided way, so that the recommended lecture tool was mistakenly used. The later tool was largely preferred by novice readers, so we conclude that the

name chosen for this tool was very confuse for them. This is a static tool that always recommends the same tour without considering the users working sessions.

Another remark was the lack of use of the guided tour tool in any tasks. We consider that the problem possibly resides in a wrongly chosen name (that poorly describes its meaning) for this tool or in the user lecture's habits (they always prefer an exploration tool).

b) Results about Expertise readers

The behavior of the more knowledgeable readers was the expected. They usually use **Key-word filters** and **Guided Tour tools** for accessing directly to the adequate nodes. They use **Navigation tool** as a complementary mechanism to fulfill the goal, never as the primary data retrieval tool.

Conclusions

We conclude that additional tools are very necessary for beginner and expert readers [Zelleweger91]. Novice users require supplementary support to avoid the problems of navigation like disorientation in the hyperspace [Simpson90], and the experts need tools that substitute navigational interaction with direct access to hypertext nodes.

Therefore, we have to analyze how to improve the wrong usage of the **Recommended Lecture Path tool**. Thus, we are conceiving two possible alternatives to solve this problem: a) An elementary solution is to change the name of the icon with a more explicit one, b) A more complex solution is to construct another tool such as a Context Recommended Lecture Path tool, that would assist the readers to chose the suitable accessible paths preventing cyclic and recurrent

ones. This tool could be required by demand or automatically when the system detects a possible user disorientation in the hyperspace.

Another problem to analyze is the seldom usage of the **Guided Tour tool** by novice readers. This tool shows the most popular nodes of the hypertext. We're creating new tests in order to investigate the sources of this user behavior and so, to propose a future improvement of this tool.

The evaluation performed will be continued in the future using not only direct observation but the log analysis of the different sessions. We understand that a continuous revision is needed mainly when using concepts so much dependent on the interaction schemes (regarding human factors & human computer interaction) as hypermedia.

References

[Sokolnicki88] : T. Sokolnicki. *Intelligent Tutoring System -Craft or Technology-*. Linkoping University, Tech Rep. LiTH-IDA-R-88-33, 1988.

[Trigg88] : R. Trigg. *Guided tours and Tabletops: Tools for communicating in hypertext environment.* ACM Transactions on Office Information Systems, 1988.

[Marshall89] : C. Marshall, P. Irish. *Guided Tours and on-line presentations: How authors make existing hypertext intelligible for readers.* Proceedings of Hypertext'89, 1989.

[Simpson90] : A. Simpson. *Lost in Hyperspace: how can designers help?.* Intelligent Tutoring Media, Vol. 1, No. 1, 1990.

[Zelleweger91] : P. Zelleweger. *Structure, Navigation and Hypertext: The status of the Navigation Problem*, Proceedings of Hypertext'91 Conference. San Antonio, Texas, December, 1991.

[Guinan92] : C. Guinan, A. Smeaton. *Information Retrieval from Hypertext Using Dynamically Planned Guided Tours.* Proceedings of Echt'92, 1992.

[Frei92] : H. Frei, D. Stieger. *Making Use of Hypertext Links when Retrieving Information.* Proceedings of Echt'92, 1992.

[Belkin92] : N. Belkin, W. Croft. *Information Filtering and Information Retrieval: Two Sides of the Same Coin?.* Communications of ACM, Vol. 35, No. 12, December 1992.

[Ichimura93] : S. Ichimura, Y. Matsushita. *Another dimension to hypermedia access.* Proceedings of Hypertext ' 93 Conference. Seatle, Washington, November 1993.

[Díaz94a] : Javier Díaz, C. Queiruga, M. A. Schiavoni. *Uso empírico de herramientas de hipermedia para la organización de dominios complejos.* Memorias de CLEI - PANEL '94 (XX Conferencia Latinoamericana de Informática). México, Septiembre de 1994.

[Díaz94b] : Javier Díaz, C. Queiruga, M. A. Schiavoni. *A proposed extension to the hypermedia model for training in networking technology.* Proceedings of BIWIT'94 (Basque International Workshop on Information Technology). Biarritz, Francia, February of 1994.

[Rivlin94] : E. Rivlin, R. Botafogo, B. Schneiderman. *Navigating in Hyperspace: designing a structure-based toolbox.* Communications of the ACM, Vol. 37, No. 2, February of 1994.

[Díaz95] : Javier Díaz, C. Queiruga, M. A. Schiavoni. *MHEF: Modelo Hipertextual con Estadísticas y Filtros.* Proceedings of Conferencia Internacional de la Sociedad Chilena de Computación, Arica, Chile, Noviembre 1995..

[Petre95] : Marian Petre. *Why looking isn't always seeing: readership skills and graphical programming.* Communications of the ACM, Vol. 38, No. 6, June of 1995.

[Kara95] : Kara, Pernice, Mary Bedth Budler. *Database support for usability testing.* Interactions, January of 1995.

Um Sistema de Autoria para Construção de Tutores Hipermídia

Paulo Pavel[1]; Gilda Helena B. de Campos[1]
Mario Cesar Abreu[2]; Ana Regina Rocha[1]; Alvaro Rabelo Jr.[2]

[1] Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro - COPPE
Caixa Postal 68511
21945-970 - Rio de Janeiro - RJ
e-mail: darocha@cos.ufrj.br

[2] Unidade de Cardiologia e Cirurgia Cardiovascular
Universidade Federal da Bahia
Fundação Bahiana de Cardiologia
Rua Augusto Viana s/n - Canela
40140-060 - Salvador - BA

RESUMO

Este artigo apresenta **Sisautor**, um sistema de autoria para construção de tutores hipermidia cujo objetivo é facilitar a autoria de tutores na área específica de Cardiologia. O sistema foi desenvolvido no contexto do projeto *Tutores Hipermidia em Cardiologia* da Unidade de Cardiologia e Cirurgia Cardiovascular da Universidade Federal da Bahia/Fundação Bahiana de Cardiologia (UCCV/FBC) em cooperação com a COPPE/UFRJ. Como exemplo de utilização do Sisautor é apresentado o tutor *Anatomia do Coração* e a experiência de seu desenvolvimento.

1. Multimídia e Hipermídia no Ensino de Medicina

Os dados utilizados pelos profissionais da saúde são multimídia na verdadeira acepção da palavra. Informações para o diagnóstico são obtidas a partir de relatórios, imagens gráficas, gráficos, tabelas, livros de referência, notas manuscritas, sequências de vídeo e/ou filme, gravações de áudio e conversas com o paciente e/ou colegas [KLEIN94].

O contato com imagens é de importância particular na formação médica. O problema, no entanto, é o acesso dos estudantes a essas imagens. As novas tecnologias permitem solucionar alguns dos problemas de acesso às imagens. Com efeito, as técnicas de digitalização permitem a obtenção de imagens de excelente qualidade. Os suportes utilizados para seu armazenamento asseguram a perenidade desta qualidade, mesmo com intensa manipulação. Outro aspecto importante é o baixo custo destes suportes, o que permite uma larga difusão dos documentos.

2. Um Sistema de Autoria para Construção de Tutores Hipermídia para Cardiologia

Atualmente é, amplamente, reconhecido o papel dos ambientes que favorecem a aprendizagem interativa, evitando que o aluno tenha uma atitude passiva de simples receptor de informações. A introdução de hipermídia no ensino pode tornar este processo ainda mais atraente e motivador. A Medicina é uma área onde a tecnologia multimídia se faz, especialmente, indicada pois como já observamos imagem e sons, são elementos fundamentais para o diagnóstico médico e para o ensino das diferentes especialidades.

Entretanto, embora se reconheça o grande interesse de se possuir tutores hipermídia, para o ensino de Medicina, os produtos de software que possibilitam a sua autoria, são de difícil utilização requerendo um longo aprendizado para a sua utilização. Isto traz, sem dúvida, dificuldades para a produção de tutores no volume e qualidade desejados. Este artigo descreve **Sisautor**, um sistema de autoria, cujo objetivo é possibilitar a construção de tutores hipermídia de qualidade, de forma fácil e atendendo às especificidades do ensino de Cardiologia.

Para atender a seus objetivos, **Sisautor** possui dois estados, o **Estado de Autoria**, onde é feita a edição das telas (fig.1) e o **Estado de Leitura**, onde é feita a consulta e avaliação e não estando disponíveis as ferramentas de edição (fig.2).

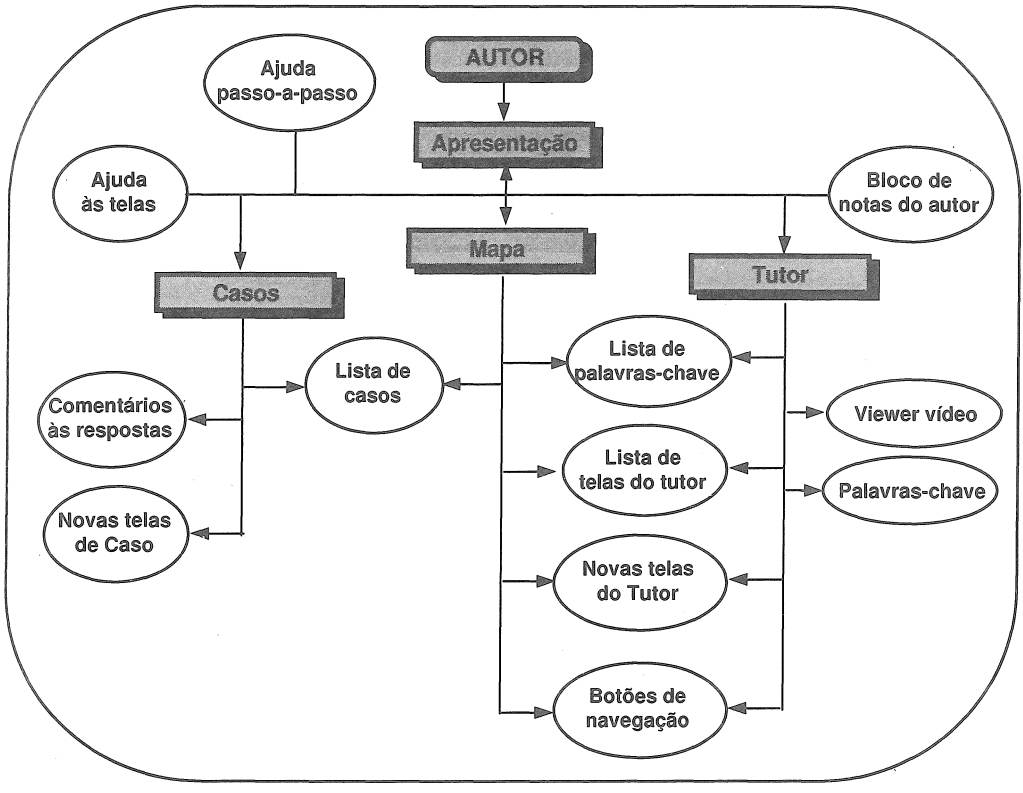


Fig. 1 - Acessos no **Sisautor** / Estado de Autoria

No Estado de Autoria é possível a navegação sem restrições, através do material já criado, e entre os quatro módulos que o compõem:

- **Módulo de Apresentação**, onde o autor edita a tela de apresentação do tutor, podendo conter gráficos e texto.

- **Módulo Tutor**, com uma tela padrão que possui um campo-texto principal, um campo para imagens e/ou gráficos, um campo base para os botões de navegação e dois campos-texto para comentários, títulos, etc. O autor pode introduzir novas telas padrão segundo sua necessidade.

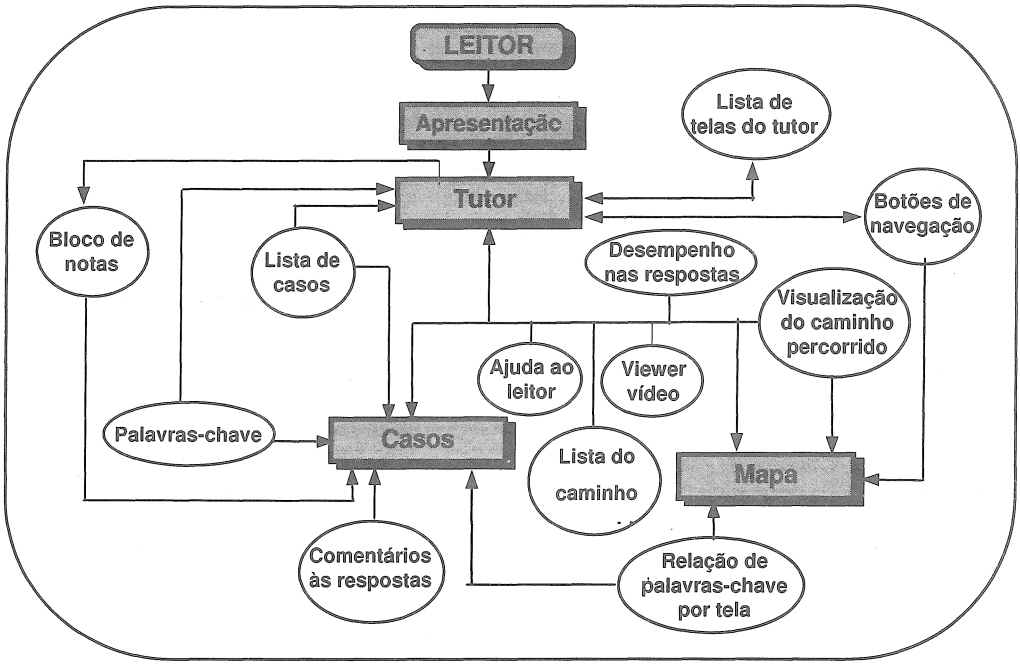


Fig.2- Acessos no **Sisautor** / Estado de Leitura

- **Módulo Casos**, com uma tela padrão que possui um campo-texto principal, um campo para imagens e/ou gráfico, um campo-texto para o título do caso, um campo de respostas com até cinco opções possíveis e campos de comentários às respostas.
- **Módulo Mapa**, onde à medida em que o autor acrescenta uma nova tela ao tutor, é criada automaticamente uma *tela-imagem* e para cada botão de navegação (criado numa tela do tutor) é ativado um *botão-imagem* na *tela-imagem* (figura 3a). Este botão mostra ao ser clicado, através de setas, a que tela o botão equivalente na tela do tutor remete (figura 3b). Mantem-se,

assim, a consistência entre a orientação à navegação e a malha de hipertexto, sem necessidade de intervenção do autor a não ser para diagramar com o mouse as *telas-imagem* na tela do Mapa. O autor pode acrescentar novas telas e botões ao tutor a partir do Módulo Mapa, assim como navegar na malha de hipertexto através das *telas-imagem*.

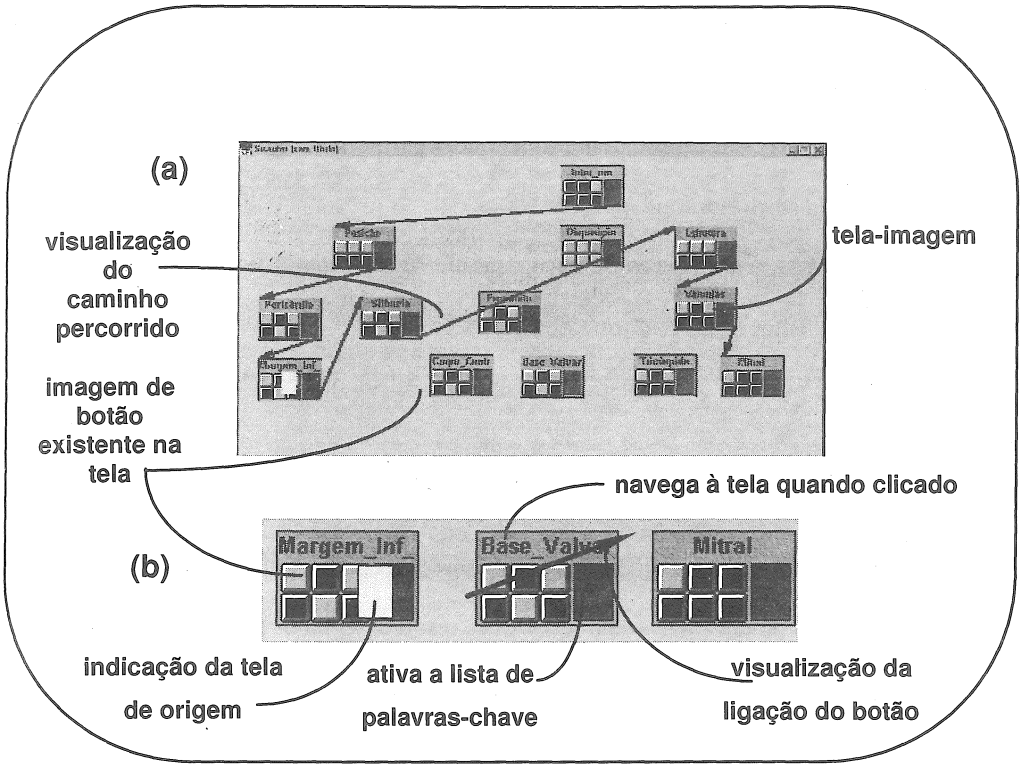


Fig. 3 - Tela do módulo Mapa do **Sisautor** (a) com a função de visualizar o caminho percorrido ativada. e (b) detalhe com a função de mostrar ligações ativada

Todas as ferramentas de edição no **Sisautor** são ativadas através de barras de botões apresentadas em janelas independentes, que podem ser desativadas e minimizadas pela ação do botão esquerdo do mouse nos botões padrão windows nas barras de título das janelas e reativadas pelo botão direito do mouse. **Sisautor** possui uma barra de botões para cada módulo que são

automaticamente apresentadas ao se entrar no módulo. A figura 4 mostra, como exemplo, a barra de botões para o módulo Tutor. Além dessas barras, tem-se uma barra com as funções do sistema (fig.5), que pode ser ativada a partir de todas as barras de botões dos módulos. Encontram-se aí, além das funções básicas, como salvar e imprimir: a navegação entre os módulos, a passagem para o modo de testes, a geração do tutor, e, a ativação dos *viewers* de ajuda (um instrui sobre os procedimentos de edição e o outro informa sobre a destinação dos campos nas telas e oferece exemplos de telas editadas). **Sisautor** possui, ainda, uma barra de apoio aos testes do tutor em construção, apresentada ao se entrar no modo de testes.

No **Estado de Leitura** é possível a navegação no Módulo Tutor e no Módulo Casos. A navegação no Módulo Tutor se dá da forma que foi determinada pelo autor ao construir o tutor. Entretanto, também, é possível a navegação através de uma lista que contém a relação de todas as telas do Módulo Tutor.

A partir das telas do Módulo Tutor o leitor pode acionar *palavras-chave* criadas pelo autor no corpo do texto, que remetem a telas do Módulo Casos, específicas para aquele contexto. Estas telas contém a descrição de um caso, que o leitor deve resolver. O leitor pode, também, escolher casos para resolução de uma das seguintes maneiras: livre escolha a partir de uma lista criada automaticamente ao serem acrescentados novos casos pelo autor, escolha randômica a partir da mesma lista e livre escolha a partir da relação de palavras-chaves disponíveis em cada tela do tutor mostrada no Módulo Mapa.

Ao optar por uma das opções de resposta, estabelecidas pelo autor e disponíveis na tela de um caso, é apresentado ao leitor o campo de comentário àquela resposta, comentando o seu acerto ou erro e automaticamente fica anotado na ficha de desempenho do leitor o título do caso e

seu acerto ou erro. O leitor dispõe, ainda, de um *bloco de notas* onde pode fazer suas anotações durante a sessão no Módulo Tutor e depois consultá-las na resolução dos casos.

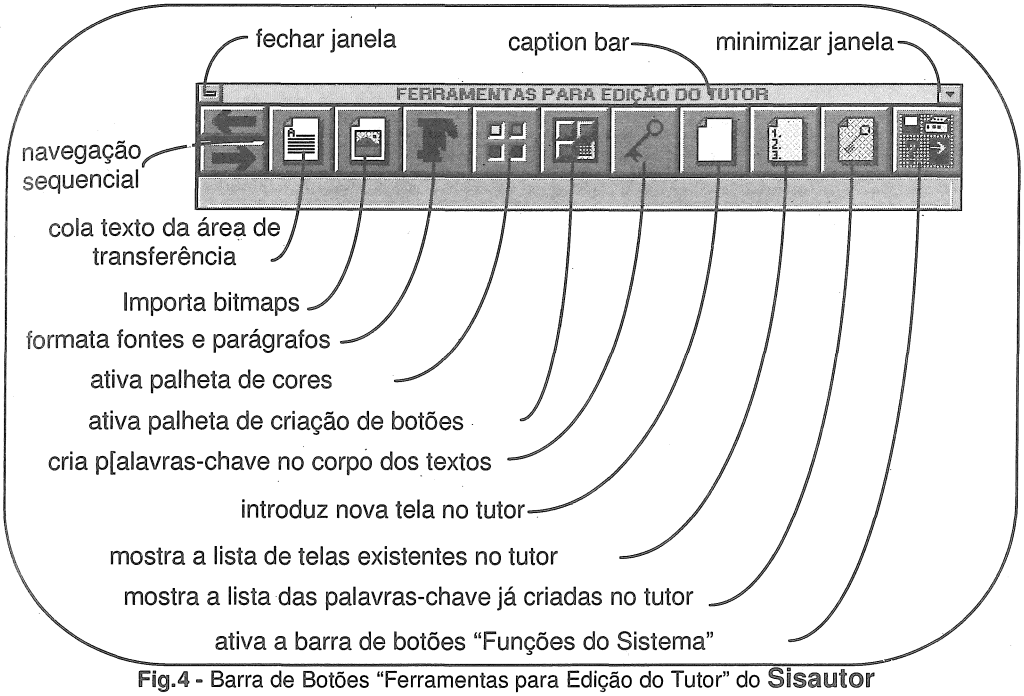


Fig.4 - Barra de Botões "Ferramentas para Edição do Tutor" do **Sisautor**

O leitor pode a qualquer momento, para sua orientação, ir ao Módulo Mapa para visualizar o caminho percorrido, consultar qual a ligação de cada botão de cada tela, consultar os casos remetidos por cada palavra-chave de cada tela, e, navegar diretamente a qualquer tela do Módulo Tutor.

Sisautor fornece ainda, para orientação do leitor, um *viewer* de ajuda (onde lhe são apresentadas informações de ordem geral, para navegação, para avaliação e sobre o tutor) e uma lista com o nome das telas já visitadas da malha hipertexto no tutor pelo leitor em ordem de

visita. A figura 6 mostra a barra de botões "Ferramentas de Controle do Sistema" do Estado de Leitura.

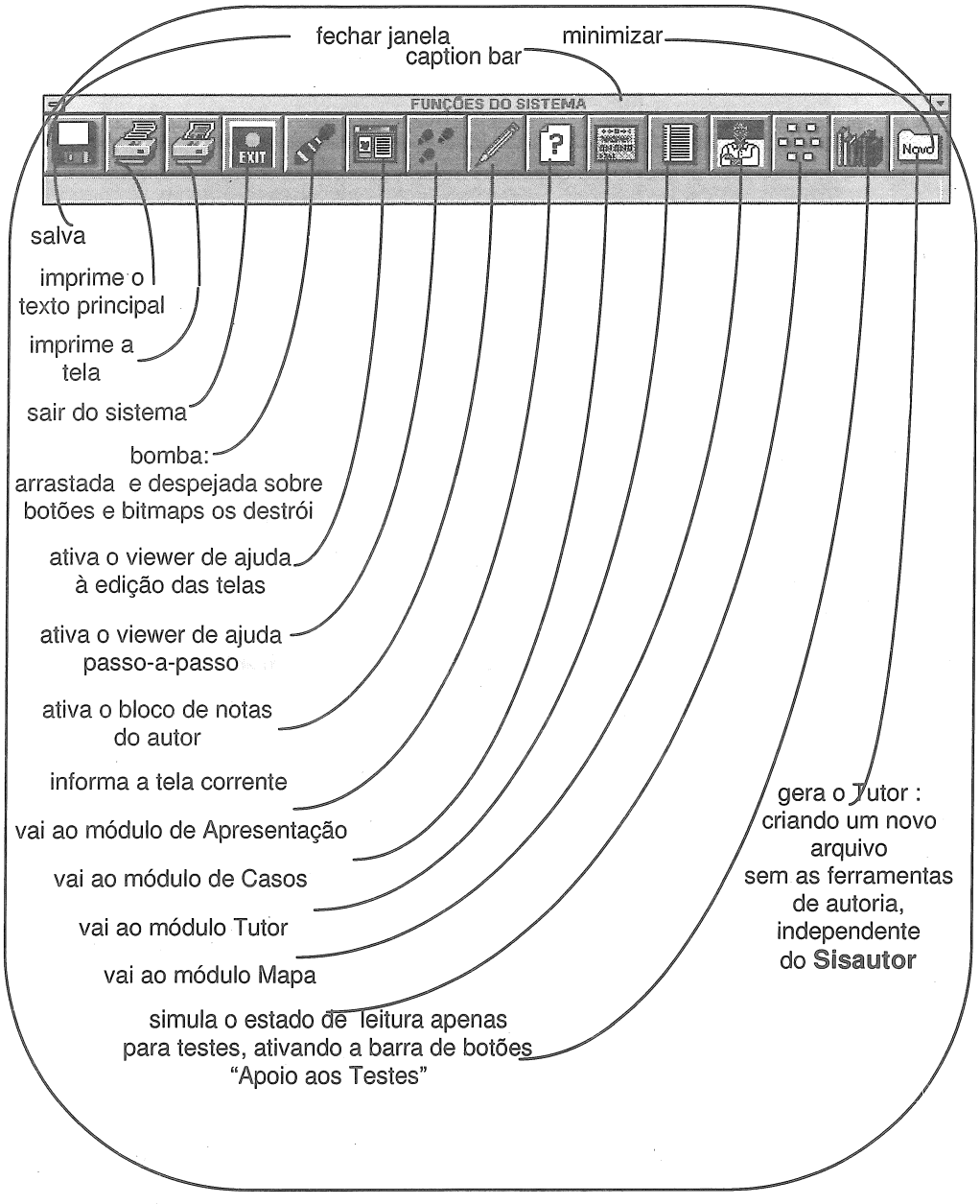


Fig. 5 Barra de Botões "Ferramentas de Controle do Sistema" do **Sisautor**

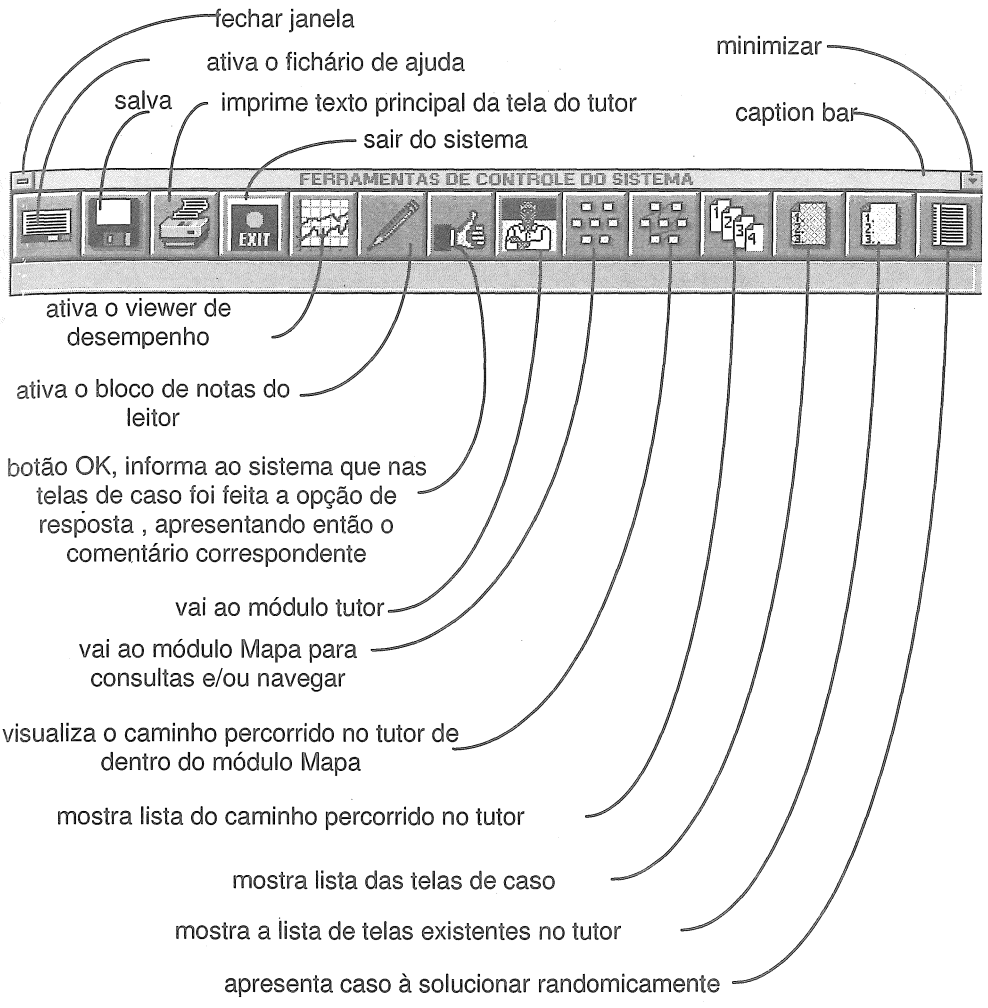


Fig. 6 - Barra de Botões "Ferramentas de Controle do Sistema" do Estado de Leitura

3. Experiência na Construção de um Tutor Utilizando Sisautor

Após a implementação do **Sisautor**, foi realizada uma primeira experiência de construção de um tutor hipermídia para cardiologia, utilizando o sistema com o objetivo de validá-lo quanto a seus objetivos. O tema escolhido foi a *Anatomia do Coração*.

O desenvolvimento de software educacional deve se dar através de equipes multidisciplinares onde o profissional de cada especialidade contribua no desenvolvimento com o conhecimento que lhe é específico. Partindo deste princípio o desenvolvimento do tutor *Anatomia do Coração* foi realizado num projeto cooperativo entre especialistas no conteúdo (dois cirurgiões cardiologistas da UCCV/FBC) e dois especialistas em Informática da COPPE/UFRJ.

Foi concebida uma tela de apresentação do tutor, como componente obrigatório, onde são fornecidas informações ao leitor sobre os critérios adotados para identificar os botões de navegação e a organização do tutor. Sendo este o único componente que dá acesso a todos os contextos, pode ser considerado um menu geral do tutor.

As telas seguiram um padrão, com as imagens fotográficas de dissecações à esquerda, e, à direita, ao lado das fotos, um desenho elucidativo, onde foram apontadas as partes anatômicas por sua nomenclatura (fig.7). Em algumas telas foram utilizadas ilustrações legendadas, para mostrar detalhes que não podem ser bem visualizados em imagens fotográficas. Outras telas, apresentam figuras com tratamento gráfico elucidativo aplicado diretamente sobre a imagem fotográfica, para ressaltar um detalhe específico mencionado no texto (fig.8). Há telas ainda, que apresentam fotos de cirurgias, mostrando o orgão vivo. Todas as telas possuem um botão para retorno ao menu principal e a navegação sequencial é permitida entre as telas de um mesmo contexto. Para navegar a outro contexto, é obrigatória a volta ao menu principal.

4. Conclusões

Este artigo descreveu **Sisautor**, um sistema de autoria para construção de tutores hipermídia para ensino de Cardiologia. **Sisautor** foi implementado no ambiente de programação Multimedia Toolbook 3.0.

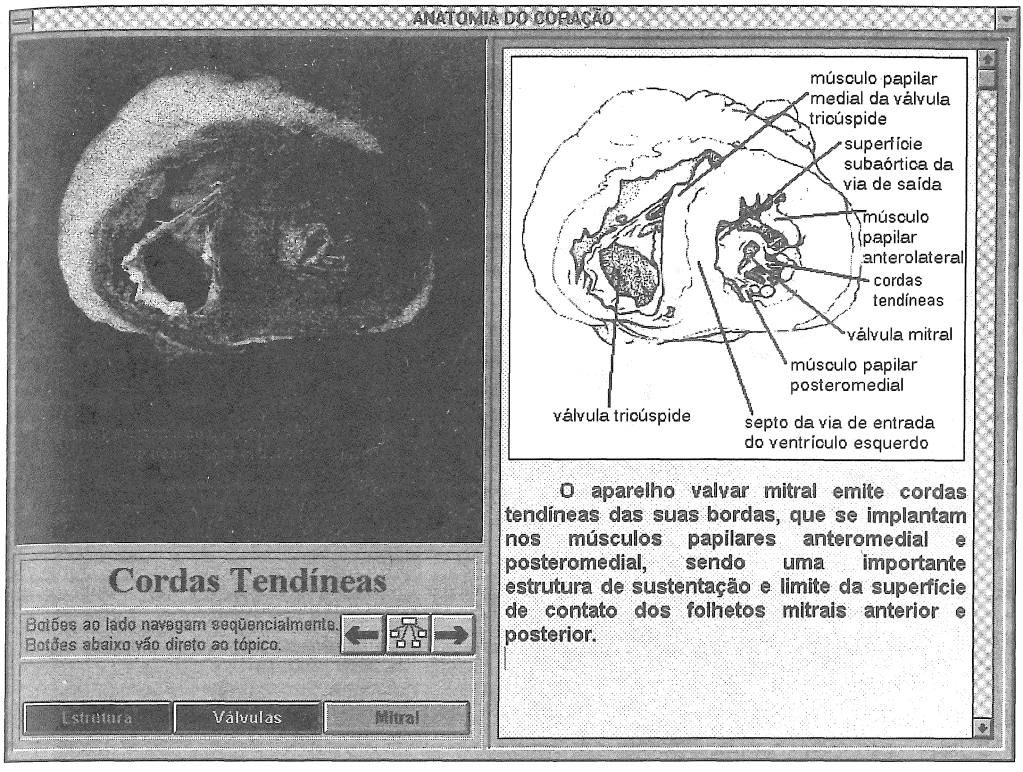


Fig. 7 - Exemplo de tela padrão apresentada no Anatomia do Coração.

Entendemos que a produção de *software* educacional é uma tarefa multidisciplinar. e foi exatamente assim que ocorreu no desenvolvimento do tutor *Anatomia do Coração*. Esta multidisciplinaridade enriqueceu e conferiu qualidade ao produto, pela ação conjunta do especialista do conteúdo (assegurou a qualidade em aspectos relativos conteúdo) e do especialista em Informática (assegurou a qualidade em aspectos relativos à Informática).

O **Sisautor** se comportou de acordo com os objetivos de sua concepção, possibilitando a construção do tutor de maneira flúida e rápida. A porção maior de tempo foi gasta na preparação das imagens gráficas e na edição dos textos. A construção do tutor propriamente dito consumiu

uma parcela de tempo que pode ser considerada mínima. Uma descrição completa do Sisautor e do tutor *Anatomia do Coração* pode ser encontrada em [Pavel 95].

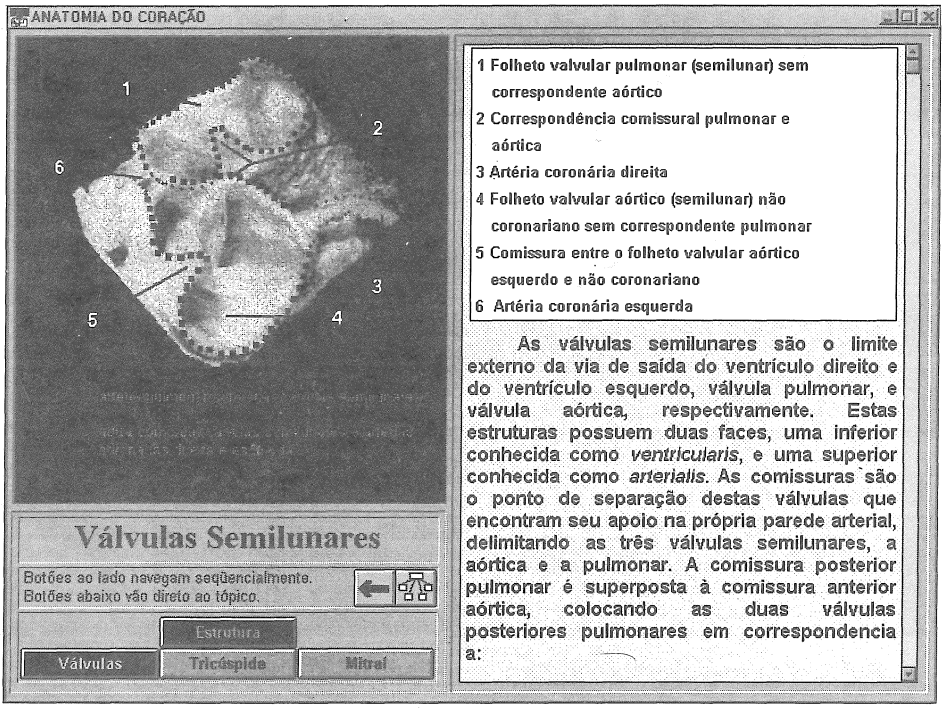


Fig. 8 - Exemplo de tela com tratamento gráfico aplicado sobre imagem fotográfica, apresentada no Anatomia do Coração.

Agradecimentos

Este projeto tem o apoio do CNPq e da IBM-Brasil aos quais os autores agradecem.

Referências

[Klein 94] Kleinholz,L; Ohly,M. Supporting Cooperative Medicine: The Bermed Project; IEEE Multimedia, vol 1, n 4; 1994

[Pavel 95] Pavel,P.C. **Sisautor: um Sistema de Autoria para Construção de Tutores Hipermídia em Cardiologia;** Tese de Mestrado; COPPE/UFRJ; 1995

Métodos de Impresión Fiel para Huellas Dactilares

Gustavo Drets, Rosario Curbelo,
Olaf Bergengruen, Andrés Almansa

Centro de Cálculo, Facultad de Ingeniería,
Universidad de la República
Montevideo, Uruguay
cecal_huellas@fing.edu.uy

29 de Noviembre de 1995

Resumen

El siguiente trabajo presenta distintas alternativas de impresión de huellas dactilares estudiadas en el marco del desarrollo del Sistema de Transmisión de Huellas Dactilares (THUEDA) realizado por la Facultad de Ingeniería para la Suprema Corte de Justicia uruguaya. Los distintos métodos de impresión permiten la reproducción de fichas decadactilares y datos asociados a bajo costo manteniendo la fidelidad de la imagen, conservando en particular los detalles necesarios para la identificación de personas. La aplicación de este sistema ha producido una aceleración substancial en el tiempo de averiguación de antecedentes, contribuyéndose de esta manera a la protección de los derechos humanos en el Uruguay.

1. Introducción

La Suprema Corte de Justicia (S.C.J.) y la Facultad de Ingeniería (F.I.) de Uruguay establecieron en 1992 un convenio de asistencia tecnológica para la informatización del archivo de fichas decadactilares y la automatización de la tarea de identificación de las mismas que realiza el Instituto Técnico Forense (I.T.F.), a consecuencia del cual la F.I. desarrolló durante el año 1994 el Sistema de Transmisión de Huellas Dactilares THUEDA [ALM95].

El I.T.F. dispone de un archivo de fichas de impresiones decadactilares registradas en papel, así como los antecedentes de todas las personas procesadas en el país desde principios de siglo, las cuales son enviadas desde los distintos juzgados por correo convencional. La demora en el envío causa, en algunos casos, un aumento en el tiempo de averiguación de antecedentes durante el cual el individuo permanece detenido, así como liberaciones de individuos sin previa consulta a los registros de antecedentes del organismo. La sustitución del envío postal por algún tipo de transmisión electrónica permite eliminar esta demora.

La S.C.J. planteó el importante problema de la fidelidad en la transmisión y reproducción de la huella ya que las soluciones existentes en el mercado no satisfacían alguno de los siguientes requisitos: ofrecer bajo costo inicial y operativo, o conservar la calidad y el tamaño de las impresiones dactilares. Debido a defectos en el entintado, rodamiento y a imperfecciones en el papel, la impresión dactilar no es uniforme, presentando zonas de escasa y excesiva concentración de tinta. Este fenómeno se ve incrementado en una reproducción a bajo costo de la ficha. A pesar de esta dificultad los métodos de impresión deben preservar la fidelidad de la misma, esto es, tanto la estructura global de los surcos como los detalles particulares.

El objetivo de este trabajo es la presentación de las distintas alternativas de impresión suministradas por el sistema THUEDA. Estas constituyen un rango de opciones desde la más sencilla, de costo computacional menor y satisfactoria en la mayoría de los casos, hasta la más compleja, de mayor costo computacional y con resultados satisfactorios incluso para huellas mal entintadas.

El presente trabajo está organizado de la siguiente manera. La sección 2 incluye las distintas alternativas estándar de impresión estudiadas, así como las deficiencias de las mismas

para esta aplicación. La sección 3 presenta técnicas de pre-procesamiento de la ficha tendientes a mejorar la reproducción de la imagen y la sección 4 plantea las conclusiones del trabajo.

2. Impresión

La imagen a ser impresa es adquirida mediante un scanner de mesa HP ScanJet IIp compatible, con una resolución de 300 ppp y 256 tonos de gris y luego es compactada con JPEG con pérdida, obteniéndose un factor de reducción del 90% en el tamaño del archivo a ser transmitido. Este factor resultó de un estudio llevado a cabo en conjunto con los técnicos dactilóscopos del I.T.F. y no afecta la clasificación ni el reconocimiento de la huella dactilar.

Un requerimiento determinante de la solución para la reproducción de la ficha es tener bajo costo inicial y operativo. Por este motivo se descartaron las impresoras por sublimación aunque proveen una fidelidad suficiente como para ser consideradas aptas para la tarea. Asimismo, cuando se diseñó el sistema, las impresoras láser de alta resolución (1200 ppp) poseían un costo igualmente elevado, además de no estar disponibles en el mercado local.

Por lo tanto se estudiaron soluciones más económicas de impresión en blanco y negro, con una resolución menor (300 y 600 ppp). La Figura 1a muestra una fotografía de una huella original, para su comparación con las restantes alternativas.

En estos casos los métodos estándar de binarización no resultan suficientes; debido a los defectos de entintado, al utilizar un umbral global a la huella se producen manchas negras o blancas que ocultan zonas considerables de la misma (Figura 1.b). A su vez los métodos de medios tonos ordenados en general utilizan patrones que a frecuencias bajas interfieren con la estructura de surcos de la huella (Figura 1.c) y a frecuencias altas producen empastado (Figura 1.d). Este problema se presenta tanto en impresoras de 300 como 600 ppp.

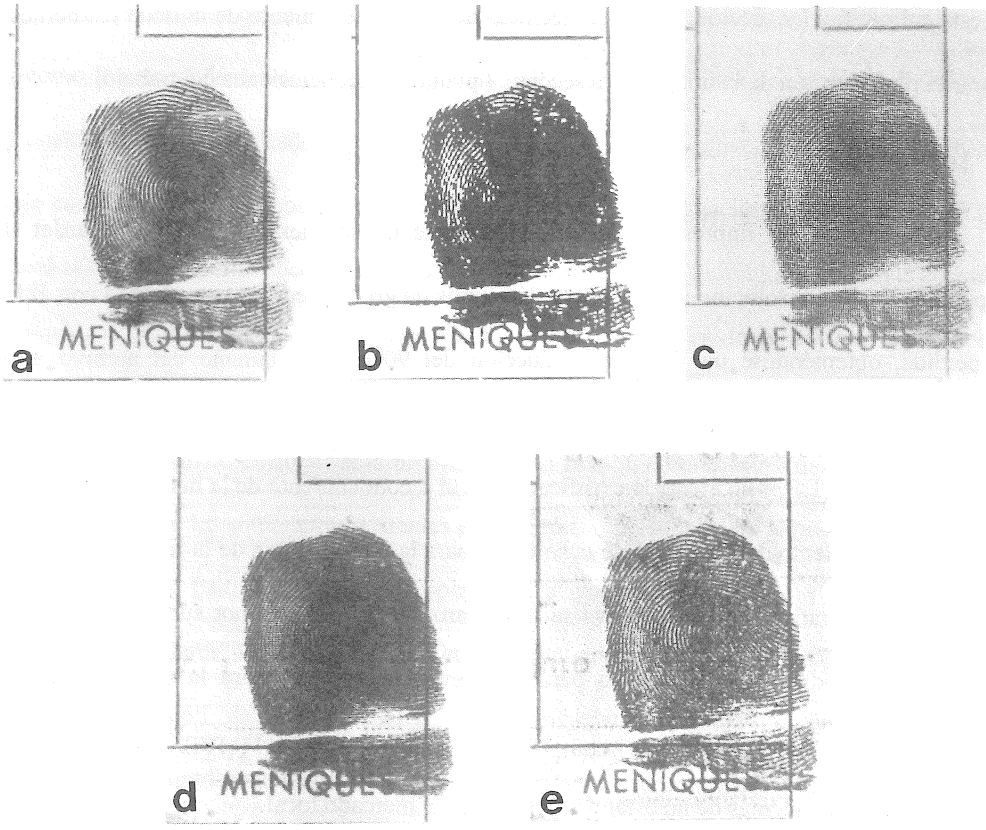


Figura 1: Alternativas de impresión: (a) Imagen original; (b) Fax (200 ppp); (c) Medios tonos ordenados a 300 ppp y 60 lpp; (d) Medios tonos ordenado a 300 ppp y 150 lpp; (e) Floyd-Steinberg (300 ppp).

Debido a las deficiencias de los métodos antes descritos se implementó la adaptación de la imagen al dispositivo de salida mediante una transformación gamma [FOL90] seguida de una binarización mediante el algoritmo de medios tonos por difusión de error de Floyd-Steinberg [FLO75] (Figura 1.e) con una resolución de 300 ppp. Para superar los resultados insatisfactorios que subsisten en las zonas de baja calidad de la imagen se agrega un pre-procesamiento de la ficha antes de su impresión.

3. Pre-procesamiento de la Imagen

El objetivo principal del procesamiento de la imagen previo a la impresión consiste en aprovechar el rango dinámico que ofrece el dispositivo de salida. En lo que sigue se analizan las distintas alternativas consideradas para lograr este objetivo.

3.1 Ecualización del Histograma

3.1.1 Ecualización Global

Uno de los métodos más conocidos para aumentar el contraste de una imagen consiste en ecualizar su histograma [GON92]. A lo largo de este trabajo se denota a la imagen original por una función $f: D \rightarrow [0, L-1]$ que puede tomar L valores distintos, cuyo dominio $D = [0, I-1] \times [0, J-1]$ es una grilla regular de $I \times J$ puntos.

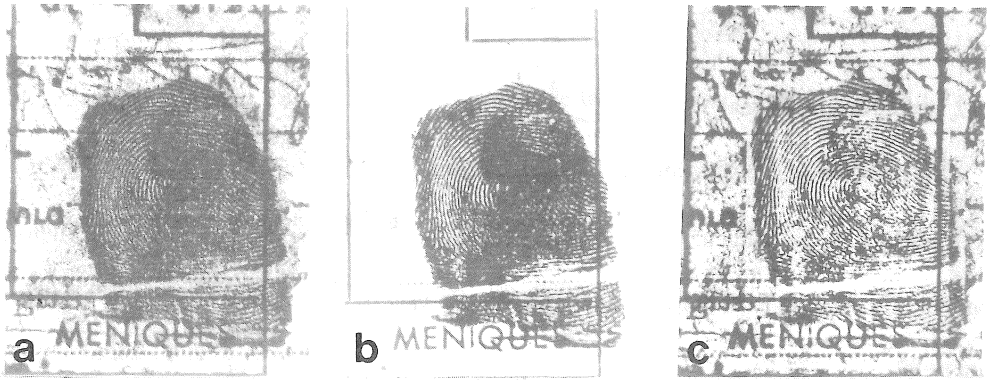


Figura 2: (a) Ecualización global clásica; (b) Ecualización del histograma global truncado; (c) Ecualización local clásica.

En el caso de las huellas dactilares, el mejoramiento debido al aumento del contraste producido por la ecualización global del histograma no es suficiente, como lo muestra la Figura

2.a. El fondo blanco de la ficha decadactilar produce un pronunciado extremo local en el histograma, que corresponde a información que no interesa realzar. Una solución consiste en eliminar el extremo local del histograma mediante un truncamiento a partir de un valor t elegido convenientemente y normalizarlo de la siguiente manera:

$$h^{(trunc)}(r) = \begin{cases} \left(\sum_{i=0}^{t} h(i) \right)^{-1} h(r) & , \text{si } r \le t \\ 0 & , \text{si } r > t \end{cases} \qquad r \in [0, L-1] \qquad (1)$$

En muchos casos la ecualización del histograma truncado produce buenos resultados, pero como se puede ver en la Figura 2.b, para la huella del ejemplo subsiste una pequeña región central donde no se distinguen los detalles. El principal problema consiste en que dentro de la misma huella coexisten regiones muy claras y muy oscuras debido a variaciones en el entintado. Para resolver el mencionado problema, es necesario efectuar una ecualización de histograma local.

3.1.2 Ecualización Local

Se realiza una ecualización del histograma local [GON92] en entornos cuadrados $E_{n,x}$ para cada punto x de la imagen. El tamaño del entorno n debe elegirse de manera de mejorar el contraste local sin realzar el ruido de la imagen. En este caso se encontró un buen compromiso para n=80.

Para lograr una implementación eficiente del algoritmo de ecualización local se tomaron en cuenta las siguientes consideraciones. En primer lugar los histogramas locales se calculan solamente en una grilla de puntos equiespaciados w pixels. Para los puntos fuera de la grilla se toma como aproximación el histograma correspondiente al punto más cercano de la grilla. Dicha aproximación es razonable siempre y cuando la relación n/w sea suficientemente grande. En este

caso una relación n/w=4 resultó ser la más adecuada. En segundo lugar los histogramas locales son calculados en forma incremental.

La Figura 2.c muestra el resultado de aplicar una ecualización local a la imagen de ejemplo donde se puede apreciar una mejor visualización de la zona oscura en el centro de la huella, sin perder los detalles de las zonas más claras. Sin embargo se observan dos nuevos problemas:

- Se realzan demasiado los detalles del fondo, lo cual produce un efecto visual desagradable.
- Los bordes de la huella se ven levemente más oscuros que el resto.

El primer problema se debe a que para dichos puntos el entorno $E_{n,x}$ contiene solamente puntos del fondo, por lo cual la transformación realza los detalles del mismo, que son irrelevantes. El segundo problema se debe a que en esos casos el entorno $E_{n,x}$ contiene, además de algunos puntos de la huella, una gran cantidad de puntos del fondo, que producen un pronunciado extremo local en el histograma para los valores de gris más claros.

La solución consiste en discriminar los puntos de la imagen que corresponden al fondo de los que pertenecen a la impresión dactilar, y efectuar un tratamiento diferenciado según el caso.

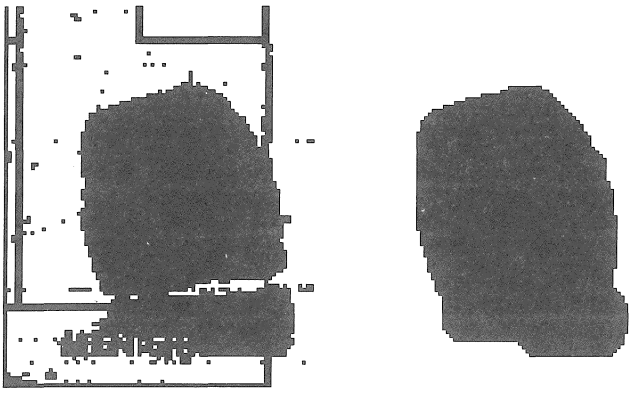
A continuación se analizará cómo calcular una máscara que permita distinguir ambas regiones y luego cómo utilizarla para mejorar el resultado.

3.2 Extracción de la Máscara

Dado que la máscara no requiere la consideración de detalles de la huella, se realiza una reducción de la resolución de la imagen a 60 ppp para disminuir el tiempo de cálculo.

La región de la huella tiene un promedio de valor de gris bajo, por lo que una primera aproximación a la máscara se realiza aplicando un filtro pasa-bajo a la imagen, binarizándola posteriormente. El umbral de binarización se calcula en función de la ubicación del máximo del

histograma de la imagen reducida. Este criterio no resulta satisfactorio en zonas muy claras de algunas huellas, donde el valor de gris de las líneas es cercano al del fondo. Para solucionarlo se agrega a la máscara los puntos tales que la variación local de tonos de gris es suficientemente alta.



(a) (b)

Figura 1: (a) Máscara antes y (b) después de la depuración

De esta manera la máscara es adecuada en las regiones de huellas, incluso para zonas de muy poco entintado, pero incluye todas las regiones oscuras de la ficha como ser líneas separadoras, letras y manchas del papel. Para eliminar estas regiones, por lo general de pequeño tamaño o espesor, se aplican a la máscara obtenida las operaciones morfológicas de apertura y clausura con un disco de 9 pixels de diámetro. La Figura 3 muestra la máscara antes y después de realizar esta depuración.

3.3 Ecualización de Histogramas Locales Modificados

Una vez obtenido el conjunto M de puntos correspondientes a la máscara se pueden resolver los problemas mencionados al final de la sección 3.1.2. En los puntos de la máscara se ecualiza un histograma calculado sólo en base a los puntos que la integran:

$$h^{(1)}_{x,M}(r) = \frac{\#\left\{y \in E_{n,x} \cap M : f(y) = r\right\}}{\#E_{n,x} \cap M} \quad r \in [0, L-1], \text{ si } x \in M \qquad (2)$$

Para los puntos del fondo no se quiere aumentar el contraste, sino dejar la imagen incambiada, lo que es equivalente al uso de un histograma constante.

$$h^{(const)}(r) = \frac{1}{L} \qquad r \in [0, L-1] \qquad (3)$$

La Figura 4.a muestra el resultado de aplicar una ecualización al histograma modificado $h^{(1)}_{x,M}(r)$ en los puntos de la máscara:

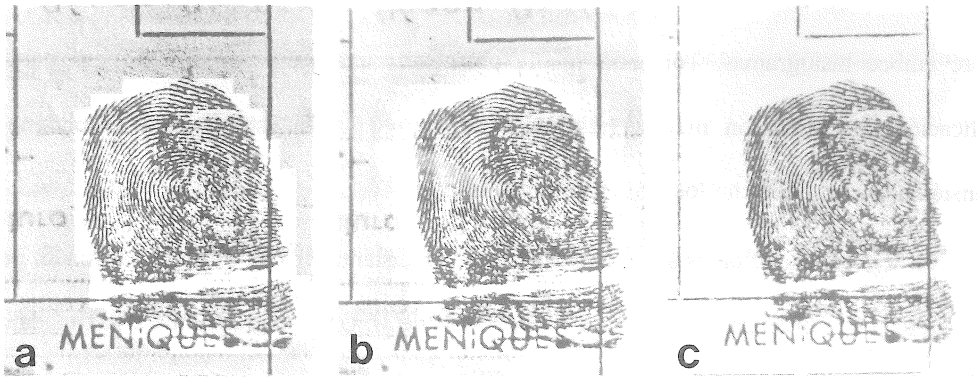


Figura 4: (a) Transición brusca, (b) Transición suave, (c) Tomando el histograma global para el fondo.

La modificación del histograma redujo el efecto visual desagradable en el fondo y evitó el oscurecimiento de los bordes de la huella, pero en cambio aparece una brusca transición desde un fondo gris fuera de la máscara hacia un fondo más claro dentro de ella. Ello se debe a la discontinuidad del histograma en la frontera de *M*.

Dicha transición brusca puede resolverse de dos maneras complementarias entre sí:

En primer lugar, la transición se puede suavizar permitiendo que el histograma varíe continuamente de $h^{(1)}_{x,M}$ a $h^{(const)}$ a medida que *x* pasa del interior al exterior de la máscara:

$$h^{(2)}_{x,M}(r) = \frac{m_x h^{(1)}_{x,M}(r) + \alpha(n^2 - m_x) h^{(const)}(r)}{m_x + \alpha(n^2 - m_x)} \qquad r \in [0, L-1] \qquad (4)$$

La imagen resultante de ecualizar $h^{(2)}_{x,M}$ refleja este comportamiento, como lo muestra la Figura 4.b.

El grado de cercanía de x a la región de huella se mide en base a m_x, el número de puntos del entorno $E_{n,x}$ que caen dentro de la máscara. El factor α permite regular el tipo de transición entre ambos histogramas. Tomando $\alpha = 1$, obtenemos una transición lineal, pero para esta aplicación se obtuvieron mejores resultados con $\alpha = 0.5$. De esta manera se obtiene una transición suave sin afectar los bordes de la huella.

La segunda solución es sustituir el histograma constante $h^{(const)}$, atribuido al fondo, por un histograma que minimiza las transiciones en el borde de M. Un histograma que aproximadamente satisface ese requisito es el histograma global calculado solamente en base a puntos de la máscara, es decir:

$$h^{(\text{global})}_M(r) = \frac{\#\{x \in D \cap M : f(x) = r\}}{\#(D \cap M)} \qquad r \in [0, L-1] \qquad (5)$$

Combinando ambas soluciones se obtiene:

$$h^{(3)}_{x,M}(r) = \frac{m_x h^{(1)}_{x,M}(r) + \alpha(n^2 - m_x) h^{(\text{global})}_M(r)}{m_x + \alpha(n^2 - m_x)} \qquad r \in [0, L-1] \qquad (6)$$

La imagen resultado de ecualizar $h^{(3)}_{x,M}$ mostrada en la Figura 4.c, si bien tiene un costo computacional superior, resuelve todos los problemas de métodos anteriores sin introducir nuevas dificultades.

4. Conclusiones

En este trabajo se presentan distintas alternativas de impresión de huellas dactilares estudiadas durante el desarrollo del Sistema de Transmisión de Huellas Dactilares (THUEDA) realizado por la Facultad de Ingeniería para la Suprema Corte de Justicia de Uruguay.

Las alternativas implementadas por THUEDA son el método de medios tonos por difusión de error de Floyd-Steinberg (sección 2), la ecualización global del histograma (1) y la ecualización del histograma local modificado (4). Para futuras versiones, se proyecta sustituir el método de ecualización local indicado por la mejora descrita en (6).

Los algoritmos desarrollados para el procesamiento e impresión de la imagen permiten utilizar equipamiento estándar y de bajo costo inicial y operativo, que de otro modo no produciría resultados aceptables. De esta manera el sistema ofrece una solución económica y eficaz al problema planteado.

Hasta el presente, se han transmitido 213 fichas con THUEDA y todas la impresiones han permitido la identificación del detenido por parte de los técnicos dactilóscopos de la S.C.J.. Esto ha significado una reducción en el tiempo de averiguación de antecedentes que alcanza, en algunos casos, hasta un 80 %, con el consiguiente beneficio tanto para el detenido como para las instituciones involucradas.

La S.C.J. de Uruguay ha manifestado interés en que el Sistema de Transmisión de Huellas Dactilares se complemente con un Sistema de Reconocimiento Automático de Huellas. El Centro de Cálculo de la F.I. comenzó estudios en esta área en 1992, los cuales se continúan desde el año

1995 con la financiación del BID-CONICYT[1] en el marco del Proyecto 96/94 titulado *"Desarrollo de un Sistema de Reconocimiento Automático de Huellas Dactilares"*. El proyecto incluye mejoramiento, binarización, detección de puntos característicos, clasificación y comparación de la imagen de las huellas dactilares. Se cuenta asimismo con la colaboración del Instituto de Análisis Numérico del Instituto Real Tecnológico de Estocolmo (NADA, KTH) particularmente a través de sus departamentos de Redes Neuronales (SANS) y Visión Artificial (CVAP). Dicha colaboración está financiada por el Consejo Sueco para Inversión y Soporte Técnico (BITS) mediante un convenio establecido entre la Facultad de Ingeniería y KTH para desarrollar proyectos tecnológicos conjuntos.

5. Bibliografía

[ALM95] Almansa, A.; Bergengruen, O.; Curbelo, R.; Drets, G. (1995) "Sistema de Transmisión Remota de Huellas Dactilares", Memoria del 5to. Congreso Internacional de Nuevas Tecnologías de La Habana, INFORMATICA'96, Cuba.

[FLO75] Floyd, R. W., y Steinberg, L. (1975). *"An Adaptative Algorithm for Spatial Gray Scale"*, International Symposium Digest of Technical Papers, Society for Information Displays, 36.

[FOL90] Foley, J. D., et al. [1990]. *"Computer Graphics"*, (Selecting Intensities-Gamma Correction pp. 564-568; Halftone Approximation, pp. 568-574).

[GON92] Gonzalez, R. C., y Woods, R. E. (1992) *"Digital Image Processing"*, (Histogram Processing, pp. 173-180; Local enhancement, pp. 182-185).

[1] Banco Interamericano de Desarrollo- Consejo Nacional de Investigación Científica y Tecnológica

A Class of Linear Algorithms to Process Sets of Segments *

Gonzalo Navarro Ricardo Baeza-Yates

Department of Computer Science
University of Chile
Blanco Encalada 2120
Santiago - Chile
{gnavarro,rbaeza}@dcc.uchile.cl

Abstract

We address the problem of efficiently performing operations on sets of segments. While current solutions to operate segments focus on single operations (e.g. insertion or searching), we are interested in set-oriented operations (e.g. union, difference and others more specific to segments). In those cases, extending the current approaches leads to $O(n \log n)$ time complexity to manipulate sets of n elements. We show that a wide class of operations can in fact be performed in $O(n)$ time, i.e. in a constant amortized cost per processed segment. We present the general framework and show a number of operations of that kind, depicting and analyzing the algorithms. Finally, we show some applications of this technique.

*This work has been supported in part by FONDECYT grants 1940271 and 1950622.

1 Introduction

In many practical applications the problem of manipulating segments arises under different forms. Typical examples are computational geometry [14, 11, 6, 3], temporal databases [7, 2], database models with constraints [9, 8] and structured text search [13, 10].

Because of this situation, the problem of manipulating a set of segments has been extensively studied [14, 5]. All these approaches focus on "single" operations, in which a single segment is operated against a set of segments. Examples are: searching for a segment, inserting a new segment into the set, removing a segment from the set, etc. In [5], it is shown how to achieve $O(\log n)$ behavior for these operations.

However, for some applications, that type of operations are less important, while more "set-oriented" operations are needed. This is the case, for example, of set-oriented query languages for structured text or temporal databases. Examples of the operations that are commonly performed in these applications are retrieving segments including other segments (e.g. all chapters containing at least four figures, in structured text search), or all segments from a set shortly preceding a segment from another set (e.g. all musical scenes where a given actor appears that shortly precede a colorful scene, in a temporal database describing movies).

A trivial extension of the current approaches to deal with these requirements leads to $O(n \log n)$ solutions for n elements. Our aim is to show that under quite general assumptions, much better solutions can be found.

We show that in many cases, a solution similar to list merging [1] can be applied to sets of segments, thus leading to $O(n)$ algorithms. Therefore, we extend this simple mechanism to deal with a more complex data structure, and analyze under which situations the idea works.

An important consideration is that we support nesting in the segments forming a set, since if no nesting is allowed, the segments are trivially linearly ordered, and it is easy to develop linear algorithms [4].

The main contribution of this paper is a simple technique to perform $O(n)$ set operations in $O(n)$ amortized time, while the cost of an individual operation is known to be $\Omega(\log n)$ in the worst case. An earlier but more detailed version of part of this work and its application to structured text retrieval can be found in [12, 13].

This paper is organized as follows. In section 2 we explain our model of operation. In section 3 we explain the general scheme of our solution. In section 4 we show a number of problems for which we found linear algorithms. In section 5 we show some complex operations for which we found more expensive solutions. In section 6 we show some applications using these algorithms. Finally, in section 7 we present our conclusions and future work directions.

2 Preliminaries

A *segment* is a pair $\langle x, y\rangle$, where x and y are real numbers and $x \leq y$. We define $From\langle x, y\rangle = x$ and $To\langle x, y\rangle = y$. A segment $a = \langle x, y\rangle$ is said to *contain* another one $b = \langle x', y'\rangle$ (and we denote it as $a \supseteq b$ or $b \subseteq a$) iff $x \leq x' \wedge y' \leq y$. If a segment contains another one we say that they *nest*. The segments are said to be *disjoint* iff $y < x' \vee y' < x$ (we say $a < b$ in the first case and $a > b$ in the other). If two segments do not nest and are not disjoint we say that they *overlap*. Finally, we use the equal sign between segments with the obvious meaning, and $a \subset b$ or $b \supset a$ to denote $a \subseteq b \wedge a \neq b$.

Our general problem is: *given two sets of segments A and B, obtain a new set C by processing A and B in some way, in $O(|A| + |B|)$ time.*

To achieve that goal, we impose two further restrictions, one on the sets and one on the processing operations:

- Each set must form a *hierarchy*, i.e. no overlaps are allowed between any two segments of a given set. This is because the only way we could obtain linear algorithms for sets with overlapping segments was preventing nesting inside the set. In this case the segments can be trivially ordered by their first or last extreme, and the normal list merging algorithms work [4]. However, in some applications (e.g. structured text search) nesting can be more important than overlapping, so we are interested in providing nesting.

- We are interested in set-oriented operations, to which we impose the additional restriction of operating with *proximal* segments. Proximality means that the presence or absence of a given segment in the final result must be defined in terms of relatively close segments in the arguments. This is because we plan to traverse the arguments in synchronization to produce the results.

Observe that we do not need that the union of both operands forms a hierarchy, as long as the result does. That is, our algorithms work also if segments from different arguments overlap.

Our approach to the solution can be defined in general terms as follows:

- We use a tree data structure to arrange the segments of each set. Since there are no overlaps, we define the tree by the containment relations between segments.

- To obtain the solution set (which is also arranged in a tree), we are going to traverse both operand trees simultaneously, in a synchronized way, while we generate the solution tree at the same time. The idea is to generalize the list merging algorithms, to make them operate on trees and select elements under more complex criteria. To be able to generate the whole solution by traversing the operands just once, it is necessary the assumption of proximal segments. All the algorithms consist of variations of this idea.

- We are not interested in how the sets are built in the first place, and how they are finally used. Our scheme is not so efficient to build the trees by consecutive random insertions, but in the applications we are interested in, those problems are solved in an ad-hoc way (we show an example in section 6).

In what follows, we describe more in detail our data structure and algorithmic scheme.

3 A Solution Scheme

3.1 Data Structure

As said, we arrange the set of segments in a tree. The criterion to define the tree is straightforward: a segment a descends from another one b in the tree iff $a \subset b$. Although for clarity we do not allow repetitions, the algorithms are easily modified to account for this.

A formal definition of our type $Tree$ follows:

$$Tree = Subtree^*$$
$$Subtree = Segm \times Tree$$
$$Segm = \{\langle x, y\rangle / x, y \in \mathcal{R} \ \wedge x \leq y\}$$
$$\forall(s, ((s_1, t_1), ..., (s_k, t_k))) \in Subtree, \forall i \in 1..k, s \supseteq s_i$$
$$\forall(s_1, t_1), ..., (s_k, t_k) \in Tree, \forall i \in 1..k-1, s_i < s_{i+1}$$

where $\mathcal{R}$ is the set of real numbers. As it can be seen, the root of our tree does not have an associated segment. Our trees can be seen in fact as forests with order among their trees.

We define some functions to access this tree type:

- $node : Subtree \rightarrow Segm$ and $subtree : Subtree \rightarrow Tree$ are selectors, i.e. if $S = (s, ((s_1, t_1), ..., (s_k, t_k))) \in Subtree$, then $node(S) = s$ and $subtree(S) = ((s_1, t_1), ..., (s_k, t_k))$.
- $head : Subtree^+ \rightarrow Subtree$ returns the first element of the list, i.e. $head(\{l_1, ..., l_k\}) = l_1$.
- $tail : Subtree^+ \rightarrow Subtree^*$ eliminates the first element, i.e. $tail(\{l_1, ..., l_k\}) = \{l_2, ..., l_k\}$.
- $\lambda \in Tree$ denotes an empty tree.

Although we do not consider any particular representation, the data structure for our trees must allow efficient computation of these access functions.

3.2 Algorithmic Scheme

The general form of our algorithms consists of traversing both trees, normally in pre or postorder. At each step, the current nodes of both trees are compared, and depending on the results and the operation, we move in one or both trees, going to the next node or jumping directly to the next sibling (thus skipping the current subtree). Each particular case is a variation of this general idea. We show a number of examples in the next section.

Thus we are going to describe the algorithms by marking nodes that must be selected or rejected from the final solution (almost all useful operations select elements from only one operand). This marking can be a simple boolean mark or it may have a more complex meaning. This provides a good abstraction, since we do not detail how we collect marked or delete unmarked nodes. Moreover, the marking algorithm can be used for two complementary operations, depending on whether we select or reject the marked nodes. Some algorithms can be solved without marking, though.

We describe now an important abstraction related to the way we traverse the trees. Unlike list traversal, which is trivial, we use two types of traversal operations here, one to "go right" and other to "go down" in the tree. Since several trees are traversed at the same time and it is wasteful to store pointers to parents, we keep explicit stacks to implement these two traversals.

- At the beginning of each algorithm, we initialize an empty stack for each argument. This stack holds pointers to $Trees$, and is referred to as $\langle argument\rangle.stack$. We use the normal operations $Push$, Pop, $Empty$ and Top on the stacks
- After initializing the stack, if the tree is not empty, we push the first top-level node of the tree into the stack.

- The algorithms access only the top of the stacks and terminate when a stack is empty. Argument tree names are uppercase letters. Their lowercase version denotes the top nodes of their corresponding stacks, e.g. $p = head(Top(P.stack))$. If $P.stack$ is empty, p is assumed to be $\langle\infty, \infty\rangle$ (where $\infty = \infty$, $\infty > x, \forall x \in \mathcal{R}$, and $\infty + x = \infty, \forall x \in \mathcal{R}$).
- To go down in the tree, we test whether the current top of the stack has children or not. If it does, we push its first child into the stack. If it does not, we perform a "go right" operation. In fact, going down means "process first the children and then the siblings".
- To go right, we replace the current top of the stack by its next sibling. If there is no next sibling, we pop the current top and retry the operation with the parent. We eventually empty the stack in this way.

We use a pseudocode notation for our algorithms. We include a `case`-like instruction (a big left brace). Figure 1 shows the basic algorithms for tree traversal.

```
Init(P)
Empty(P.stack).
If (P ≠ λ) Push(P.stack, P).
Down(P)
If (subtree(p) ≠ λ) Push(P.stack, subtree(p))
else Right(P).
Right(P)
While (¬Empty(P.stack) ∧ tail(Top(P.stack)) = λ) Pop(P.stack).
If (¬Empty(P.stack)) Top(P.stack) ← tail(Top(P.stack)).
```

Figure 1: Basic operations for tree traversal.

We use the following numbers in the analysis: n_X is the size of the set corresponding to operand X, h_X is the height of its tree (in the worst case it can be n_X) and d_X is the maximum degree of its tree (it can also be n_X in a flat tree). We also use n, d and h as the maximum corresponding value between all operands (there are normally two operands).

Two observations about the analysis:

- It should be clear that either collecting marked or deleting unmarked nodes is $O(n)$ time, where n is the number of nodes of the tree.
- Although a particular operation of tree traversal can work up to $O(h)$, we note that the whole traversal, even by using `Down`, is $O(n)$. This is because no edge of the tree is traversed more than twice. So, the amortized cost of tree traversals is always linear with the size of the tree.

We are now in position to describe a number of example algorithms. Their description is very simple with these primitives.

4 Linear Operations

There are a number of interesting problems that admit a linear implementation, for example: set manipulation, segments including or included in others, segments after or before others, etc. In this section we explain in detail a couple of them and their analysis.

4.1 Set Difference and Intersection

Set difference and intersection are complementary versions of a single marking algorithm. The idea is that the trees are unmarked at the first place, and we traverse both trees in synchronization, marking all nodes of the first argument that are also in the second one. Thus, we later implement set difference by collecting unmarked nodes and set intersection by collecting marked nodes.

Figure 2 shows the algorithm to mark the tree. P and Q are the arguments.

```
Init(P).  Init(Q).
While max(To(p), To(q)) < ∞
  p < q :  Right(P).
  p > q :  Right(Q).
  p = q :  Mark p.  Down(P).
  p ⊂ q :  Down(Q).
  else :   Down(P).
```

Figure 2: Marking algorithm for set difference or intersection.

This algorithm is linear, since a single traversal is done on each argument (that traversal has been already shown linear), and we work $O(1)$ at each step. Therefore, we have $O(n_P + n_Q) = O(n)$ time. It is also $\Omega(n)$ in the worst case.

4.2 Segments Included in Others

Another interesting operation is $\texttt{In}(P, Q)$, that selects elements from P that are included in a segment of Q.

The algorithm to solve $\texttt{In}(P, Q)$ is presented in Figure 3. We traverse the top levels of P and Q, in synchronism. When a node of P is included in a top-level node of Q, that subtree of P is marked. Otherwise we discard that P node and continue with its children.

To avoid too much marking overhead, we state that a mark in a node means that not only that node but also its whole subtree is considered marked. The collection algorithm must be modified to account for this.

This algorithm is $O(n_P + n_Q) = O(n)$, by the same arguments as before. In this case, we can refine this analysis as follows: each time we do a $\texttt{Down}(P)$ is because it contains an element of Q or because it overlaps with an element of Q, thus each extreme of each segment of (the top-level of) Q is compared, at most, with a complete path of P (length h_P). That is because once we descend the first level in P, the relevant list from the top-level of Q has only one element (the original q). At each level of this path, the operation can take us d_P comparisons, thus the cost is $O(d_Q h_P d_P) = O(d^2 h)$. That means, for example, that in an application with constant d and balanced trees the operation takes $O(\log n)$ (at least to do the marking).

```
Init(P).  Init(Q).
While max(To(p),To(q)) < ∞
  { p < q :  Right(P).
    p > q :  Right(Q).
    p ⊂ q :  Mark p.  Right(P).
    else :   Down(P).
```

Figure 3: Marking algorithm for segments included in others.

Then, the complexity of this operator is $O(\min(n_P + n_Q, d_Q h_P d_P)) = O(\min(n, d^2 h))$.

5 Complex Operations

Although most operations can be implemented in linear time, there are more complex operations that seem not to have a linear implementation: segments included in others with positions, segments including k segments, and complex versions of "after" and "before" are some examples. We explain in detail the first one.

5.1 Segments Included in Others with Positions

An operation that happens to be useful for applications is to select segments from P that are included in some segment q of Q *at a given position.* The position is defined in terms of the other segments of P that are included under the same q (e.g. the third section of each chapter). Since there may be segments included in q that nest in P (e.g. sections inside sections), we define "position" by only considering the maximal segments of P included in q, for each Q. If a segment of P overlaps with q, the segment and its descendants in P are not considered (see Figure 4). Other criteria produce slightly different algorithms.

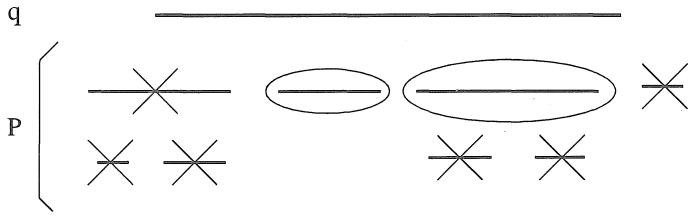


Figure 4: Criterion to participate in inclusion with positions. Ellipses indicate selected segments.

Another concern is which language will we use to denote the allowed positions of an included node. We can use, e.g. first, k-th, last, last$-k$, prime positions, etc. For this algorithm we use a generic predicate to avoid any restriction: s is an expression denoting the allowed positions.

The algorithm requires simple boolean marking. We traverse both trees in synchronization. When we find a set of nodes of P included in one of Q, we mark the s-th nodes, and then we pass

again over the included P-nodes, this time comparing them with the subtree of the Q-node. If, instead, the node of P includes one of Q, we follow the children of the P-node. Figure 5 shows the algorithm.

```
Init(P).  Init(Q).
While max(To(p), To(q)) < ∞

  p < q :  Right(P).
  p > q :  Right(Q).
  p ⊂ q :  lp ← Top(P.stack).   pos ← 1.
           While head(lp) ⊂ q
              If (pos ∈ s) Mark head(lp).
              lp ← tail(lp).   pos ← pos + 1.
           Down(Q).
  else :   Down(P).
```

Figure 5: Marking algorithm for segments included in others at a given position.

To analyze this algorithm, consider that we can traverse both P and Q completely, and that the final collection of nodes is $O(n_P)$. Observe that each element of Q is deleted from the problem by doing at most $O(d_P)$ work (when $p \subset q$), thus the algorithm is $O(n_P + n_Q d_P)$. But also observe that each element of P can be worked on by at most a complete path of Q, thus the algorithm is also $O(n_Q + n_P h_Q)$. Then, the algorithm has the best from both complexities, namely $O(\min(n_P + n_Q d_P, n_Q + n_P h_Q)) = O(n \min(d, h))$.

To see that it is also $\Omega(n^2)$, consider the following example: $\texttt{In}(\{\langle 1,1\rangle, \langle 2,2\rangle, ..., \langle n,n\rangle\}, \{\langle 1,n\rangle, \langle 2,n\rangle, ..., \langle n,n\rangle\}, s)$.

This non-linear complexity may be surprising, since the requirement of proximality seems to be met. But observe that in order to determine whether a P node is to be marked or not, we have to consider many nodes of Q, not just one, and there seems to be no better way to do that. If the language of positions (s) is restricted we can obtain better complexities, e.g. if we allow to express only ranges we can implement it in $O(n \min(h, k \log d))$ where k is the number of ranges we have.

6 Some Applications

Our technique applies to a number of dissimilar applications. In this section we briefly discuss a temporal database and explain in detail a structured text search application. Our aim is not only to expose real situations where the problem arises, but also to show the practical performance of our algorithms.

6.1 Temporal Databases

Temporal databases [7, 2] can manipulate information with temporal validity. Their data is based on events that occur at a given point or interval in time. If we have a set-oriented query language we will be interested in questions such as "give me all the events (and their times) that satisfy some constraint" (an example is given in the introduction).

In fact, temporal databases are much more complex, since they also deal with uncertainty in time (e.g. A happened before B, without knowing when). We just want to show that if a database has to manage a number of quantitative facts (i.e. events that are known to have happened at known times), it may be interesting for its back-end to perform set-oriented operations on these facts. These algorithms would be a part of a more general inference engine, being an efficient way to select relevant facts to work on.

In most temporal databases, the time intervals can overlap and nest. As explained earlier, we cannot handle results (final or intermediate) having segment overlapping. Therefore, this is a restriction we impose on the applicability of our model to temporal databases.

This does not mean that there cannot be overlaps in the database. We can divide the knowledge into a set of sequential processes, and subsets of different processes can be combined into operations, although each answer and intermediate result is a subset of some sequential process. Those sequential processes can be hierarchically structured.

6.2 Structured Text Search

Modern textual databases are a good example of hierarchical structuring. We describe here a specific model to query structured text databases, called Proximal Nodes [13, 12]. The development of this model has motivated this work on efficient algorithms to evaluate queries, although some operations differ slightly from those exposed here.

In this model, a textual database is seen as a text (a sequence of symbols) plus a number of independent hierarchical structures built on the text. Each hierarchy is strict, although there may be overlaps between different hierarchies. The nodes of the hierarchies are the structural components of the database (e.g. chapters, pages, etc.). Examples of hierarchical views of the text are: chapter/section/paragraph and fascicle/page/line. In this case, it makes full sense for answers to be a subset of a given hierarchy.

The leaves of the query syntax trees are names of structural components and text pattern-matching expressions. Each structural component (e.g. chapters, figures, pages) is preindexed, so the set of all elements of a given type can be retrieved in a time proportional to the size of the retrieved set. Pattern-matching uses another index to return a list of segments of the text that matched the pattern. These lists are considered to be part of a special hierarchy. This is how the sets are built in the first place. All internal nodes of query syntax trees correspond to operations between segments of the operands, and they are further restricted to operate only on proximal segments.

It is shown that the query language obtained under these restrictions has good expressivity, while the implementation (using the algorithms we expose here) is very efficient. The model constitutes a good tradeoff between the models which achieve high efficiency by strongly restricting the structuring possibilities and the query language of the database and those which have rich structuring and querying facilities but a high execution overhead.

Some examples of queries are:

- I want text paragraphs in italics which are before (but in the same page) a figure that says something about the earth.
- I want a paragraph preceding another paragraph where the word "Computer" appears before (at 10 symbols or less) the word "Science". Both paragraphs must be in the same page.
- Give me all references to Knuth's books in chapters 2-4.

- I want all sections with mathematical formulas that are not appendices.

We also implemented a lazy version of the algorithms, that only processes the nodes that are needed for the final result (thus saving a lot of work when processing the intermediate results). This mechanism can also be used to implement a navigational interface, in which the user sees the top-level of the result tree and asks to expand only some nodes. In this case, we save the work of computing nodes that are not to be seen. Although the worst-case complexity of the lazy operations is worse than that of "full" operations, the real times are better, since less nodes are processed in practice.

A prototype has been implemented for this model, to test the average performance of the algorithms. We used a database of C programs and Latex documents, whose structuring is quite different. We tested each operation on different operands of sizes (i.e. number of nodes) ranging from 100 to 10000. We also tested a number of more complex queries, to compare the full and lazy versions on real queries.

Figure 6 shows a typical example of the times of an algorithm (i.e. a single operation). They correspond to a Sun SparcClassic of approximately SpecMark 26 and 16 Mb of RAM. From the tests we extract the following conclusions (they may differ for other applications):

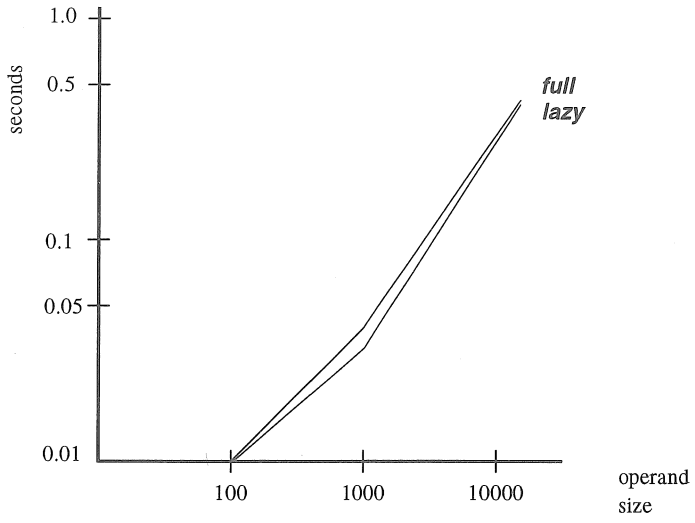


Figure 6: Typical times for equal-sized operands. Observe that we use a logarithmic scale.

- The full versions of the algorithms are all linear in practice, since the situations under which they are not are very unlikely to occur in structured text databases (e.g. a very deep tree).
- The full versions have very low variance, being their times highly predictable, proportional to the sum of the sizes of all intermediate results. The constant for our machine is approximately 50.000 nodes processed per second per operator.

- The lazy version is normally better than the full one in practice, despite the worse complexities, especially for complex queries. This is because less nodes are expanded. Lazy algorithms have very large variance, though.

7 Conclusions and Future Work

We analyzed the problem of manipulating a set of segments when we are interested in set-oriented operations. Classical solutions provide simple operations with $O(\log n)$ complexity, which leads to $O(n \log n)$ solutions for set-oriented operations.

However, we show that, under some general assumptions, set-oriented operations can be performed in linear time. These assumptions are: the operands and the result must not have overlapping segments inside, and the operations must work on proximal segments.

We developed a framework oriented to tree traversal that generalizes the idea of list traversal for merge-like operations, and applied the framework to solve in linear time a number of set-oriented example operations on segments. The aim was to show that the framework is flexible enough to be adapted to a number of apparently dissimilar problems. This technique works well for set-oriented operations because of the amortized cost, i.e. its performance for single operations is not good.

Finally, we presented some applications to which this technique could be applied. We also include average times of a real implementation of the algorithms.

There are a number of future work directions related to this work. The most important are:

- Extend these ideas to allow overlaps into a set, since this will open a wealth of new possibilities to apply this technique.
- Find a technique that keeps these good results while improving the performance for single operations.
- Search for more operations that can be implemented in linear time by using this framework, specially by looking at applications that can benefit from this work.
- Study a disk-based implementation, trying to minimize seek times. Some work on this direction has already been done.
- Extend the framework to account for more dimensions, e.g. manipulating hypercubes instead of just one-dimensional segments. This may open a number of opportunities of applications in computational geometry and in map processing (e.g. in geographic information systems).
- Study a parallel implementation, since our algorithms seem to be highly parallelizable.

Acknowledgments

We thank Claudia Medeiros and Nina Edelweiss for their help on temporal databases. We also thank the helpful comments of the referees.

References

[1] A. Aho, R. Sethi, and J. Ullman. *Compilers: Principles, Techniques and Tools.* Addison-Wesley, 1986.

[2] J. Allen, J. Hendler, and A. Tate, editors. *Readings in Planning.* Morgan Kaufmann, 1990.

[3] J. Bentley. Algorithms for Klee's rectangle problems. Dept. of Computer Science, Carnegie-Mellon Univ. Unpublished notes., 1977.

[4] C. Clarke, G. Cormack, and F. Burkowski. Schema-independent retrieval from heterogeneous structured text. In *Procs. of the 4th Annual Symposium on Document Analysis and Information Retrieval*, Apr. 1995.

[5] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[6] H. Edelsbrunner. Dynamic data structures for orthogonal intersection. Technical Report F59, Tech. Univ. Graz, Institute für Informationsverarbeitung, 1980.

[7] A. T. et al. *Temporal Databases: Theory, Design and Implementation.* Benjamin Cummings, 1993.

[8] P. Kanellakis and D. Goldin. Constraint programming and database query languages. Technical Report CS-94-31, Dept. of Computer Science, Brown University, June 1994.

[9] P. Kanellakis, S. Ramaswamy, D. Vengroff, and J. Vitter. Indexing for data models with constraints and classes. Technical Report CS-93-21, Dept. of Computer Science, Brown University, May 1993.

[10] A. Loeffen. Text databases: A survey of text models and systems. *ACM SIGMOD Conference. ACM SIGMOD RECORD*, 23(1):97–106, Mar. 1994.

[11] E. McCreight. Priority search trees. Technical Report CSL-81-5, Xerox PARC, 1981.

[12] G. Navarro. A language for queries on structure and contents of textual databases. Master's thesis, Dept. of Computer Science, Univ. of Chile, Apr. 1995. `ftp://sunsite.dcc.uchile.cl/-pub/users/gnavarro/thesis95.ps.gz`.

[13] G. Navarro and R. Baeza-Yates. A language for queries on structure and contents of textual databases. In *Proc. ACM SIGIR'95*, pages 93–101, 1995. `ftp://sunsite.dcc.uchile.cl/-pub/users/gnavarro/sigir95.ps.gz`.

[14] F. Preparata and M. Shamos. *Computational Geometry.* Springer-Verlag, 2nd edition, 1988.

Cálculo Aproximado de Diagramas de Voronoi utilizando Autómatas Celulares

Christian Trefftz
Departamento de Informática y Sistemas
Universidad EAFIT
Medellín Colombia
ctrefftz@sigma.eafit.edu.co

Abstract

Se presenta un modelo que utiliza autómatas celulares para resolver en forma aproximada el problema de los diagramas de Voronoi.

1 Introducción

El problema de construir diagramas de Voronoi [1, 9] se puede describir así: Dado E^d (un espacio euclideano de d dimensiones) y un conjunto de puntos de entrada (sitios o generadores), encuentre el conjunto de puntos en el espacio que están mas cerca de cada generador.

Si $d = 2$, entonces el problema se convierte en la "baldozinación" (tiling) de un plano. Los diagramas de Voronoi en el plano tiene aplicaciones en muchos campos diferentes: Gráficas computadorizadas [2], segmentación por texturas [6] y navegación de robots [8].

Las primeras soluciones [4, 5] a este problema se basaron en encontrar las ecuaciones de las líneas que están a distancias iguales entre cada par de puntos,

calcular los interceptos entre estas líneas y recortar (clip) apropiadamente. Una nueva alternativa se basa en la técnica de barrido del plano [3]. Una línea barre el plano y las líneas que forman el diagrama se van calculando a medida que se va barriendo el plano.

Para ciertas aplicaciones, una aproximación gráfica es suficiente para propósitos prácticos. Al usuario le interesa una imagen que le suministre información acerca de la estructura de la "baldozinación". Una aproximación gráfica se puede generar en la pantalla de un computador.

2 Una primera aproximación

Si una aproximación es suficiente, se puede usar la definición de Diagrama de Voronoi para colorear los pixels de la imagen de acuerdo con el generador que esté mas cerca, es decir, se asigna un color diferente a cada generador, y, para cada pixel, se calcula distancia desde ese pixel a todos los generadores. Al pixel se le asigna el color del generador mas cercano. Si el pixel queda a la misma distancia de dos generadores diferentes, este pixel se encuentra en la frontera entre dos regiones del plano y se colorea negro. La imagen resultante es una aproximación del diagrama de Voronoi en el plano.

En el siguiente algoritmo se asume que cada pixel y los generadores tienen coordenadas x y y que han sido escaladas apropiadamente del dominio del problema al sistema gráfico del computador que se está usando. En este proceso se introducen errores de redondeo, pero como se dijo anteriormente, solo se está calculando una aproximación.

La forma general del algoritmo es la siguiente:

```
begin
  Lea las coordenadas de los generadores
  Para cada pixel en la pantalla
    Calcule la distancia entre el pixel y cada uno de los generadores
    Asigne al pixel el color del generador mas cercano
  FinPara
  Muestre la imagen
end
```

Este algoritmo se presta para ser implantado en máquinas paralelas en las que cada área de la pantalla se pueden asignar a un procesador diferente. Trefftz, Szakas y McKinley realizaron una implantación en un MasPar MP-1. Los detalles se pueden encontrar en [7].

Si uno considera cada pixel de la pantalla como una celda de un autómata celular, este algoritmo se puede considerar, en un sentido amplio, como una aplicación de autómatas celulares. El lector argumentará que este algoritmo no corresponde realmente a un autómata celular porque el estado de las celdas del autómata no depende exclusivamente del estado anterior de cada celda y del estado de las celdas vecinas sino de unos datos que hay que difundir a todas las celdas participantes.

3 Una segunda versión

Si se quiere ser mas fiel al "espíritu" de los autómatas celulares, los nuevos estados de los autómatas se deben basar únicamente en el estado previo anterior y en los estados de los vecinos.

Si se sacrifica la inmediatez de la respuesta, aún es posible obtener una solución. En el momento inicial, únicamente los puntos generadores tienen color y todos los demás pixels estarán "en blanco". En cada uno de los pasos siguientes, cada pixel realizará la siguiente operación: Si el pixel ya tiene asig-

nado un color, conserva su color. Si el pixel no tiene asignado un color, examine los colores de sus vecinos: Si ningún vecino tiene color, permanezca sin color; Si hay uno o mas vecinos con el mismo color, adopta ese color; si hay vecinos con colores diferentes, este pixel se encuentra en una frontera entre regiones y se colorea negro.

La forma general de esta nueva versión del algoritmo es la siguiente:

```
begin
  Cada generador tiene un color diferente
  Repetir hasta que no haya cambios
    Para cada pixel
      Si no tiene color asignado
        Examine los vecinos
        Si no hay vecinos con color,
          permanezca sin color
        FinSi
        Si uno o mas vecinos tienen el mismo color,
          adopte ese color
        FinSi
        Si hay dos o mas vecinos con diferentes colores,
          adopte el color negro
        FinSi
      FinSi
  FinRepetir
  Muestre la imagen
end
```

4 Conclusiones y Trabajo Futuro

Se ha descrito una aplicación de los autómatas celulares para obtener una solución aproximada al problema de los diagramas de Voronoi.

La experiencia con este algoritmo sugiere que los autómatas celulares son un modelo conceptual muy apropiado para ciertas aplicaciones en computadores paralelos. Planeamos explorar la factibilidad de utilizar modelos de autómatas celulares en redes de estaciones de trabajo.

Agradecimientos

El autor desea expresar sus agradecimientos a B. Ostrowsky por una muy útil conversación acerca del tema.

References

[1] F. Aurenhammer, "Voronoi Diagrams - A survey of fundamental geometric data structures," *ACM Computing Surveys*, vol. 23, pp. 345-405, 1991.

[2] David P Dobkin, "Computational Geometry and Computer Graphics", *Proceedings of the IEEE*,Vol. 80,No. 9,pp.1400-1411,1992.

[3] F. P. Preparata y M. Shamos, "Computational Geometry: An Introduction", Springer, New York, 1985.

[4] Green,P.J y R. Sibson, "Computing Dirichlet tessellations in the plane," *The Computer Journal*,21(2),168-173,1978.

[5] Kokichi Sugihara y Masao Iri, "Construction of the Voronoi diagram for "One Million" Generators in Single-Precision Arithmetic", *Proceedings of the IEEE*,Vol. 80,No. 9,pp.1471-1484,1992.

[6] Mihran Tuceryan y Anil K. Jain, "Texture Segmentation Using Voronoi Polygons," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12,No. 2,pp.211-216,1992.

[7] Christian Trefftz, Joseph Szakas y Philip McKinley, "A Parallel Algorithm for Graphically Approximating Voronoi Diagrams", Technical Report ACS-81, Department of Computer Science, Michigan State University, 1993.

[8] Ana María Trejos y Juan David Velásquez, "Diseño de una Arquitectura para un Robot Autónomo Navegante", Proyecto de Grado, Departmento de Informática y Sistemas, Universidad EAFIT,1993.

[9] G. Voronoi, "Nouvelles applications des parametres continus a la theorie des formes quadratiques," *Journal fur die reine und angewandte Mathematik*, vol. 133, pp.97-178, 1907

Modelo para la Simulación de Fenómenos Turbulentos para Ambientes de Realidad Virtual

Daniel Arturo Barrero Ulloa
e-mail: d-barrer@ uniandes.edu.co

José Tiberio Hernández
e-mail: jhernand@uniandes.edu.co

MOX - Computación Avanzada en Ingeniería
Universidad de los Andes

Abstract

El modelaje de fenómenos turbulentos complejos y su influencia sobre diversos elementos a los cuales da forma como el humo, fuego, nubes, etc., es un problema complejo y a la vez atractivo debido a la aplicabilidad de su solución en la vida practica. Su solución requiere la combinación de campos de movimiento, flujo gaseoso, e iluminación realista. Aquí se presenta un modelo por medio de niveles turbulencia para los campos de movimiento, cada uno de los cuales se puede definir de manera diferente, siendo cada nivel afectado únicamente por el nivel anterior, esta subdivisión permite contornear las discontinuidades de las ecuaciones de dinámica de fluidos, facilitando su solución. Además se utiliza un método de modelaje por medio de bases de turbulencia para los niveles, basado en la observación de formas características en los fenómenos de este tipo, y en el cual se introducen irregularidades por medio de modelaje estocástico. Para el modelo de los gases y su animación se utiliza un modelo de avección-difusión embebido en un campo de movimiento.

Categorías y Descriptores de Tema: **[Computación gráfica]**: Gráficas Tridimensionales y Realismo, Generación de Imágenes, **[Mecánica de Fluidos]**: Turbulencia, Dinámica de Fluidos, Fenómenos Gaseosos, **[Probabilidad y Estadística]**: Procesos Estocásticos, Algoritmos Probabilisticos.
Frases y palabras clave adicionales: flujo turbulento, campos de movimiento, modelaje estocástico, humo, fuego, modelaje por capas, modelos de iluminación.

Introducción

Al observar diversos fenómenos naturales como el humo, el fuego, las nubes, se aprecia cómo la forma de estos objetos está caracterizada por irregularidades que son una de sus principales rasgos distintivos. En el caso de los fenómenos gaseosos como el humo, estas irregularidades son resultado de la interacción del objeto con el medio, y sus alteraciones en la forma son realmente la visualización de los llamados fenómenos de turbulencia.

Para el modelaje de los fenómenos de turbulencia se requiere que se modele no solo el objeto en cuestión (p.ej. humo), sino que se debe modelar también el medio (viento), y la interacción entre los dos (el fenómeno en si). Este problema puede ser atacado de maneras muy diversas: desde el modelo físico completo del fenómeno lo que se trata por medio de dinámica de fluidos [FOL92][KAS90][ROG84][CFD95][STR95], pasando por modelos

estocásticos[AVE89][FOL92][STA93][REE85][WIT87], hasta modelos puramente geométricos[EBE94][FOL92][HAR86] en los que no existe un modelo implícito del fenómeno.

Lo que se desea es encontrar un modelo para este tipo de fenómenos, que no solo sea una buena aproximación, sino que también pueda ser realizado de una manera suficientemente eficiente, de modo que se pueda ofrecer realismo en las imágenes producidas para sistemas de tiempo real (Realidad Virtual, etc.). Así mismo se quiere encontrar un modelo que sea controlable, y suficientemente flexible para poder permitir su combinación con otros métodos de manera que se puedan lograr niveles distintos de realismo de acuerdo con diferentes necesidades.

El modelaje estocástico es una alternativa muy natural para este tipo de fenómenos, ya que se puede simular la aparición de irregularidades de manera controlada y es lo bastante flexible para poder permitir su aplicación en una amplia variedad de fenómenos[CFD95][STR95].

El modelo propuesto aquí difiere de los propuestos anteriormente, en el sentido que se utiliza el modelo estocastico sobre un modelo geométrico de los fenómenos para modelar diferentes niveles de turbulencia. Otra diferencia es el que tradicionalmente se supone la evolución del tiempo como una constante deterministica (hipótesis de Taylor), mientras que aquí corresponde también a un proceso estocastico. En el modelo presentado se manejan el medio como campos de movimiento divididos en componentes de gran y pequeña escala, y los gases como distribuciones de densidad de partículas, cuya evolución se describe también por medio de procesos estocásticos.

I. Bases de Turbulencia

Si se observan con detenimiento los fenómenos turbulentos, se pueden identificar claramente un conjunto de formas las cuales aparecen de manera repetitiva, como los vórtices, que aparecen no solo de manera aparentemente independiente, sino que muchas de las formas observadas en estos fenómenos se pueden producir a partir de la combinación de un conjunto de geometrías base.

De acuerdo con esto un fenómeno de turbulencia se puede asimilar a un grupo cuya base corresponde a un conjunto de campos de movimiento linealmente independientes característicos del fenómeno, los cuales están modelados de acuerdo con la geometría de las irregularidades observadas (que llamaremos formas de turbulencia), así como una serie de operaciones sobre éstos. En otras palabras un fenómeno de turbulencia se puede definir como:

$$Ft = <\mathbf{ft}, \mathbf{Op}>, \mathbf{ft} = \{ft_1, \ldots ft_n\}, \mathbf{Op} = \left\{Op_1, \ldots, Op_m \therefore Op_i : \mathbf{ft} \times \mathbf{ft} \xrightarrow{Op_i} \mathbf{ft}\right\} \quad (1)$$

Donde Ft es el fenómeno de turbulencia, Op es el conjunto de operaciones posibles sobre la base, y el conjunto ft es el conjunto de formas de turbulencia, tales que ft_i, $1<i<n$, corresponden a un conjunto de formas de turbulencia linealmente independientes.

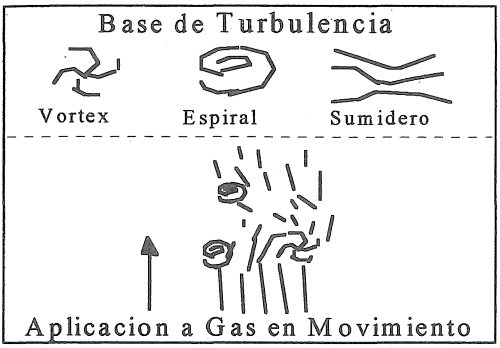


Fig-1. Ejemplo Base de Turbulencia 2D

Cada forma de turbulencia ft_i corresponde a un campo de movimiento de manera tal que éste está definido en el espacio por medio de una función dependiente de la velocidad y presión del fluido. Como se trata de aprovechar el conocimiento de la geometría del fenómeno, entonces el campo se puede definir por medio de funciones vectoriales en el espacio, que representan la geometría, son continuas, y en los lugares donde no se encuentra definida se supone que el fluido no es afectado. De esta manera, las operaciones que se realizan para la composición de las diferentes formas de turbulencia se reduce a la composición de funciones vectoriales simples.

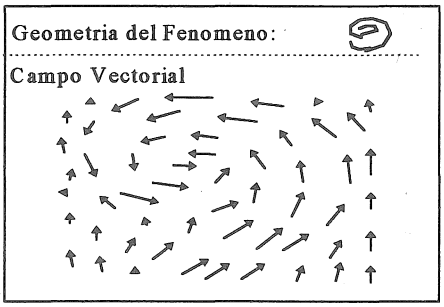


Fig-2. Ejemplo de forma de turbulencia.

Como un campo de movimiento está definido por las variaciones en la velocidad del fluido, y la función de la forma de turbulencia $f(\mathbf{x},t)$ solo indica la trayectoria a seguir por el fluido, entonces las variaciones en la velocidad del mismo están dadas por la segunda derivada de la función. Si a esto además se le adiciona una tasa de disipación k debida a la interacción entre los diferentes modos de velocidad en el campo ocasionado por los cambios de presión, entonces tenemos la siguiente ecuación:

$$\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t} = \nabla^2 f(\mathbf{x},t) - k * \nabla p(\mathbf{x},t)$$

o lo que es igual:

$$\frac{\partial u(x,t)}{\partial t} = \nabla u(x,t) - k * \nabla p(x,t) \quad (2)$$

Esta ecuación corresponde a un solo campo de los pertenecientes a una base de turbulencia, y tiene como ventaja el que se puede resolver fácilmente por métodos numéricos tradicionales, de manera bastante eficiente debido a que es una ecuación diferencial lineal de primer orden.

De acuerdo a lo anterior se introduce la definición de campos base como aquellos que es posible calcular por medio de una función especifica, ya sea Navier-Stokes, Campos Aleatorios, o en el caso de las bases de turbulencia, un campo definido por medio de la ecuación (2).

II. Modelaje de Múltiples Niveles para Campos de Movimiento

La forma de un gas está dada principalmente por la manera como éste interactua con el medio en el que se desarrolla, es por eso que se debe modelar de alguna manera las irregularidades presentes en el medio que son las que finalmente van a determinar la evolución del gas, estas irregularidades corresponden a los fenómenos de turbulencia. Si se observa con detenimiento el comportamiento de un medio, se puede ver que es fácil realizar una correspondencia directa entre este y un campo vectorial (aquí llamados campos de movimiento). En el caso de los gases estos campos de movimiento son llamados más tradicionalmente campos de viento.

Físicamente, la turbulencia es un conjunto de fenómenos producidos por diversas razones: Obstáculos en el camino del fluido, viento, cambios de presión, etc. Estos fenómenos producen cambios en la velocidad, dirección, y presión del fluido en el espacio-tiempo, esto se representa tradicionalmente en la dinámica de fluidos por medio de las ecuaciones de Navier-Stokes[STA93][ROG84][CFD95][STR95], que se pueden resolver por medio de diversos métodos numéricos.

Esta aproximación para el cálculo de los campos de movimiento, adolece de varios problemas: En primer lugar, es complicado lograr el efecto deseado por medio de la aplicación de fuerzas externas exclusivamente; y en segundo lugar, las no-linealidades presentes en las ecuaciones las hacen difíciles de resolver numéricamente, especialmente en presencia de turbulencia. Linealizar estas ecuaciones puede mejorar bastante la estabilidad y eficiencia[CFD95][STR95][15], pero puede llegar a disminuir drásticamente la aparición de fenómenos de turbulencia (como en el caso de fluidos viscosos).

En la dinámica de fluidos tradicional, se observa que un campo de movimiento esta compuesto por dos niveles de turbulencia, a pequeña y a gran escala. Basándonos en esta

observación se propone un modelo para los campos de movimiento de múltiples niveles extendiendo esta definición de la siguiente manera:

Se supone que un campo de movimiento no base u(x,t), se puede definir como la composición lineal de los dos conjuntos de campos posibles, (gran escala (U_g(x,t)) y de pequeña escala (U_p(x,t))):

$$u(x,t) = \alpha U_g(x,t) + \beta U_p(x,t), \quad (3)$$

Donde cada una de los conjuntos de campos U_g y U_p corresponden a su vez a composiciones de campos de movimiento así:

U_g es una composición lineal de campos base:

$$U_g(x,t) = \sum_{i=1}^{n} \alpha_i u_{b_i}(x,t), \quad (4)$$

U_p es una composición aleatoria de campos a pequeña escala $u_i(x,t)$:

$$U_p(x,t) = \sum_{i=1}^{n} u_i(x,t) * \sigma_i * r(x,t), \quad (5)$$

En esta ecuación, σ_i corresponde a la varianza del campo vectorial a pequeña escala $u_i(x,t)$, y $r(x,t)$ corresponde a una función de ocurrencia previamente determinada. Esta función de ruido es la que va a determinar la densidad de aplicación los diferentes campos de movimiento a pequeña escala, y es por medio de esta función, que se agregan irregularidades en la aparición y aplicación de los fenómenos turbulentos, para lograr un mayor realismo.

Cada uno de los campos a pequeña escala $u_i(x,t)$ corresponde a un campo vectorial no base, de manera que por la definición de campo entonces todos los u_i se definen como:

$$u_i(x,t) = \alpha U_{g_i}(x,t) + \beta U_{p_i}(x,t), \quad (6)$$

De esta manera la definición de un campo esta dada por un árbol n-ario de composición de campos con m niveles de profundidad, la definición recursiva de esta función termina cuando para todos los u_i de un nivel m determinado los U_{pi} correspondes a un conjunto vacío, es decir solo esta definido por los U_{gi} (composición lineal de campos base).

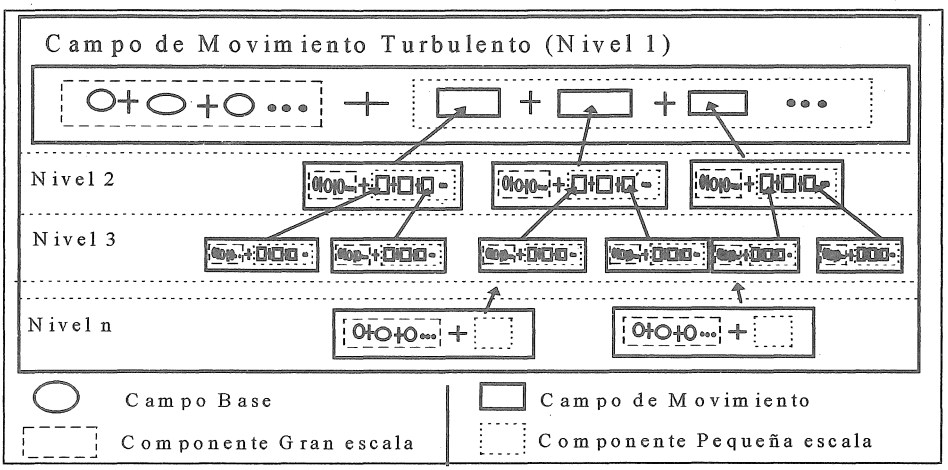


Fig-3. Modelo de Campos de Movimiento por Niveles.

III. Evolución de los Fenómenos de turbulencia en el espacio-tiempo

Tradicionalmente el manejo del tiempo en la simulación de fenómenos turbulentos se toma como una evolución temporal deterministica (hipótesis de Taylor), mientras que aquí la evolución del tiempo con respecto al espacio va a corresponder a un proceso estocástico, el cual afecta de manera directa la evolución espacial del fenómeno para cada instante de tiempo.

Como se menciono inicialmente en el modelo de un campo de movimiento turbulento depende de los cambios en la velocidad y presión, así como de fuerzas externas, pero en general un campo de movimiento se define como una función vectorial en el espacio n dimensional donde se encuentra el fluido, si a esto se le agrega el tiempo como una dimensión mas, entonces se logra una mayor libertad en el modelo, a que es posible representar una serie de funciones mas complejas, así como es posible disminuir la complejidad de las funciones de cálculos de la evolución temporal.

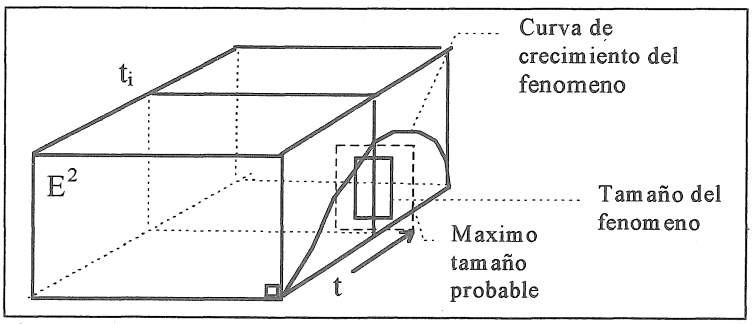


Fig-4 Evolución de un fenómeno en el Espacio-tiempo.

Al introducir el tiempo como una dimensión adicional se puede utilizar este para controlar la definición de los múltiples niveles de definición en un campo de movimiento turbulento, esto porque no solo controla la aparición de los fenómenos de manera aleatoria, sino que determina el tamaño máximo (definición) del fenómeno turbulento a pequeña escala en un instante de tiempo determinado, permitiendo así el manejo mas simple de los múltiples niveles de definición del campo.

Para manejar el tiempo se define una función de probabilidad de manera que esta permita predecir cuando es el próximo instante de tiempo en el que va a ocurrir un evento de importancia en la simulación, definiendo como evento como los cambios significativos en proceso de simulación como es la aparición de un fenómeno de turbulencia, su desaparición, cambio de tamaño, y en general los eventos de máximo impacto sobre el comportamiento de los campos de movimiento. La ocurrencia de estos eventos corresponde a encontrar los máximos y mínimos de las funciones de definición de tiempo, y campos.

Esto permite que el tiempo se maneje de manera sincrónica con la ocurrencia de eventos, de manera que en la implementación se puede aumentar la eficiencia al tener que calcular únicamente los campos solo en los instantes de tiempo determinados, y generar una interpolación de resultados entre dos eventos de tiempo sucesivos (de todas maneras cabe anotar que la evolución de un fenómeno de turbulencia en un instante dado de tiempo depende de su estado en el instante anterior).

Por lo que la ecuación 5, se puede reescribir así:

$$U_p(\mathrm{x},t) = \sum \mathrm{u}_i(\mathrm{x}, T_i(t)) * \sigma_i * r(\mathrm{x}, T_i(t)) + \sum u_i(\mathrm{x}, t-1) * T_i(t) * \sigma \mathrm{t}_i, \qquad (7)$$

En donde $T_i(t)$ es una distribución de probabilidad en el tiempo, la cual se encarga de definir la evolución de una forma especifica de turbulencia en el tiempo, dando los cambios máximos de tamaño de esta desde su inicio, hasta el instante en que desaparece. El segundo termino corresponde a una distribución de probabilidad dependiente del estado del campo en el instante anterior con la inclusión de una función de varianza para modificar el tamaño de los fenómenos presentes en el campo (fig-4).

III. Modelo de un Gas

Físicamente un gas corresponde a un conjunto de partículas, pero esto requiere de una gran cantidad de partículas, para lo cual se necesitaría un poder computacional bastante grande para su manejo (en el caso que este sea posible). En cambio resulta mucho mas practico el hecho que todo sistema de partículas tiene asociada una función de densidad en el espacio-tiempo $\rho(x,y,z,t)$, la cual puede estar definida de manera deterministica o estocastica, y se encuentra definida para todo el espacio, cuya evolución esta dada por una ecuación de Avección-Difusion (A-D)[AVE89][STA93][15] a la cual se le ha adicionado un termino de disipación $\alpha(\rho,t)$dependiente del tipo de gas a modelar:

$$\frac{\partial \rho}{\partial t} = -u\nabla\rho + \kappa\nabla\rho^2 - \alpha(\rho, t) \tag{8}$$

El primer termino en el lado derecho es el termino de avección, el cual es el encargado de involucrar los efectos del movimiento del gas, asumiendo que el gas tiene un vector de movimiento **u** dependiente del tiempo. El segundo termino calcula la difusión molecular a una rata κ. Y el tercer termino corresponde a la disipación de la densidad.

Se asume además que las partículas no tienen efecto en el medio, por lo que todo el movimiento de las mismas esta dado exclusivamente por su localización y por el efecto que este tenga sobre las mismas. De tal manera que es la combinación de esta función de densidad con los diferentes modelos de los fenómenos de turbulencia la que va a simular la interacción del gas con el medio determinando así la evolución del mismo a lo largo del tiempo que es lo que en resumen nos interesa.

IV. Interacción Entre el modelo de Turbulencia y el de Gases

La interacción entre el modelo de un gas y el modelo de los fenómenos de turbulencia es la que al final nos permite visualizar los efecto de estos fenómenos sobre los objetos naturales, como en el humo, las nubes, etc.

En la ecuación 8 donde se define un gas como una distribución de densidad, se descubre que basta con reemplazar las funciones de velocidad y presión por las definidas por el campo de movimiento (u y p), con el que interactua el gas. Pero si bien esto produce resultados de manera muy directa, la solución a esta ecuación para campos complejos, se vuelve rápidamente prohibitiva tanto en tiempo computacional como en memoria. Entonces para poder obtener los mismos resultados sin este problema se puede realizar de la siguiente manera[STA93][15]:

Se supone que la distribución de densidad es una suma balanceada de una distribución simple *f*:

$$\rho(\mathbf{x}, t) = \sum_{i=1}^{n} m_i(t) * f(\|\mathbf{x} - \mathbf{x}_i(t)\|, t - t_i) = \sum_{i=1}^{n} \rho_i(\mathbf{x}, t), \tag{9}$$

En otras palabras la distribución de densidad es una "burbuja difusa", con una función de campo dependiente del tiempo *f*, donde $\mathbf{x}_i(t)$ es el centro de masa, t_i es el tiempo al que la "burbuja" ρ_i es creada, y $m_i(t)$ es su masa. Si se supone que *f* es una distribución gausiana con una desviación estándar σ_0 mucho mas pequeña que la escala mas pequeña en el campo de movimiento, el campo se puede suponer constante en cada burbuja.

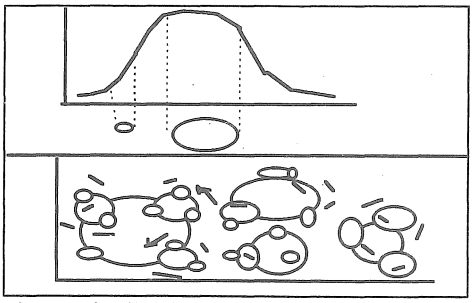


Fig-5. Distribución de densidad como burbuja difusa,
(El tamaño se asocia con una discretizacion de la distribución de probabilidad).

Entonces el termino de aveccion solo mueve la burbuja, pero no deforma su forma, por lo que el movimiento del la burbuja esta dado por la integral de su centro de masa sobre el campo de viento:

$$\mathbb{x}_i(t) = \mathbb{x}_i(t_i) + \int_{t_i}^{t} \mathrm{u}(\mathbb{x}_i(s), s)ds, \qquad \mathrm{i}=1,\ldots,\mathrm{n}, \tag{10}$$

Y la deformación de su forma esta dada por el termino de difusión. Aquí cabe notar que la difusión a una tasa κ después de un tiempo $t\text{-}t_i$ de una gausiana con varianza σ_0^2 es equivalente a la convolución de una gausiana con varianza $\kappa(t\text{-}t_i)$ con una gausiana de varianza σ_0^2. Como las gausianas son cerradas bajo convolución[AVE89][15] la gausiana resultante tiene una varianza $\sigma_i^2(t) = \sigma_0^2 + \kappa(t - t_i)$, y:

$$f(r, t - t_i) = \frac{1}{(2\pi)^{\frac{3}{2}}\sigma_i^3(t)} \exp\left(-\frac{r^2}{2\sigma_i^2(t)}\right), \tag{11}$$

Entonces f se difumina hacia afuera con una varianza $\sigma_i^2(t)$ que aumenta con t. El factor de normalización $(2\pi)^{\frac{3}{2}}\sigma_i^3(t)$ garantiza que la masa de la burbuja es invariante bajo difusión. Una vez la varianza de una burbuja se vuelve comparable a la escala mas pequeña del campo de movimiento entonces se puede reemplazar por burbujas mas pequeñas y distribuir la masa igualmente entre ellas. El efecto del termino de disipación es una disminución exponencial de las masas sobre el tiempo:

$$m_i(t) = m_0 * \exp(-\alpha(t - t_i)), \tag{12}$$

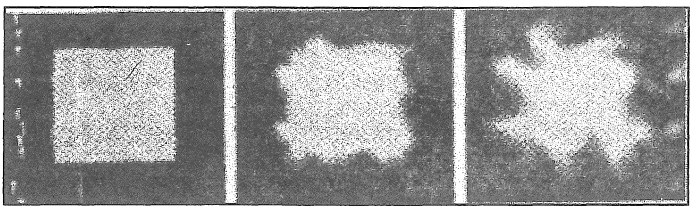


Fig-6. Evolución de una Distribución de Densidad

V. Modelo Computacional de un Sistema para el modelaje y simulación de fenómenos turbulentos.

V.1. Consideraciones del Diseño del Sistema

Con base en la fundamentacion teórica presentada se propone un modelo computacional para un sistema para el calculo de estos fenómenos, basándose en consideraciones como:

- Desempeño: Como lo que se desea es el poder obtener imágenes que muestren estos fenómenos de manera bastante creible para poder aumentar la sensación de realismo en imágenes generadas por computador, y como también se desea que los resultados se puedan aplicar en aplicaciones interactivas, entonces se debe reducir al máximo la complejidad de las ecuaciones a resolver, también debe ser posible controlar el nivel de definición de los resultados, y la suavidad de la simulación, así como el poder cambiar las características del modelo de turbulencia si esto mejora el desempeño. Esto se logra mediante la utilización de bases de turbulencia, y de manejo de tiempo por medio de procesos estocástiscos, así como por medio del modelo de múltiples niveles.
- Ortogonalidad: Como se desea poder cambiar las características del modelo de turbulencia, debe ser posible que en el caso que esto se realice no ocurran efectos colaterales en otras características, como por ejemplo cambiando el modelo de un nivel de bases de turbulencia a dinámica de fluidos.
- Interoperabilidad: Debe ser posible el integrar este sistema de simulación de estos modelos con otros sistemas tanto de simulación como con diferentes sistemas de visualización, de manera que partes de este sistema se puedan incluir en otro, o viceversa.
- Completitud: Se debe poder tener acceso a toda la funcionalidad del sistema sin importar las características especificas de implementación de las diferentes partes del sistema.
- Extensibilidad: Debe ser posible extender la funcionalidad del sistema, de manera fácil, sin afectar la integridad del mismo.

V.2. Diseño del Sistema

El sistema de simulación gira en torno a la representación del estado del proceso en simulación el cual esta representado por el modelo de niveles, en cada nivel tenemos un conjunto de "burbujas" propuesto por el modelo de gases y un conjunto de definición de campos asociado a cada burbuja propuesto por el modelo de turbulencia. Por supuesto cada nivel es el resultado de la evaluación de sus niveles inferiores.

La sofisticación del proceso de evaluación depende de cuantos niveles se tomen en cuenta. En resumen, existe un anillo de procesos que, conjuntamente pueden simular el proceso de evolución de un fluido en presencia de fenómenos de turbulencia. Ellos son Modelo Turbulencia, Modelo Gas, Visualización, Modelo del Tiempo. Existen obviamente procesos de apoyo como: Solucionador de Ecuaciones (se pueden utilizar procesos externos como

programas especializados para el manejo de estos datos p.ej. mathematica, etc.), Manejo de persistencia, etc.

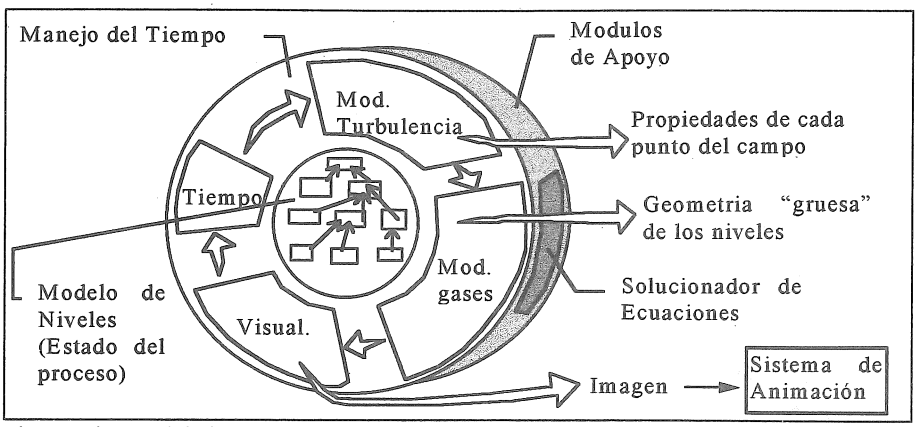


Fig-7. Diseño del sistema de Simulación (El sistema gira alrededor de la representación del estado del proceso en simulación).

VI. Visualización

Se han definido dos modos de visualización de los datos producidos por la simulación dependiendo del modelo de iluminación que se utiliza (para el nivel de realismo), así como considerando eficiencia en la obtención de los resultados. En general ambos modelos están basados sobre las técnicas tradicionales de ray-tracing [EBE94][FOL92][STA93][REE85], pero con diferentes modelos de iluminación, así como técnicas de optimizacion especiales para cada uno.

En ambos modelos se supone que el gas esta compuesto por burbujas difusas de diferentes tamaños dependiendo de la densidad del gas y de los niveles de turbulencia. El primer modelo de visualización el cual llamamos de visualización temprana o para visualización interactiva, corresponde a un modelo simple de ray-casting para la generación de las imágenes, en el cual solo se realiza un chequeo de intersección del rayo con la frontera de las burbujas, y no se contemplan fenómenos como difracción o reflexión los cuales desvían el rayo permitiendo optimizar la velocidad del algoritmo de intersección por medio de métodos de clustering [EBE94][STA93], en este modelo se incluye el fenómeno de atenuación de la luz dando un coeficiente de atenuación para cada burbuja (burbujas-semitransparentes). Este modelo genera imágenes aproximadas de manera muy eficiente, lo cual permite controlar el sistema de manera mas interactiva.

El segundo modelo de visualización o modelo de visualización final, incluye modelos de iluminación mucho mas reales, así como fenómenos comúnmente asociados con los gases como refuerzo, emisión de luz, y mejora el modelo de intersección al incluir difracción y reflexión, lo que afecta el rayo haciendo que este cambie de dirección y no se debe calcular

el efecto de la burbuja solo en la frontera, sino también en su interior. Este modelo genera imágenes bastante reales, pero a un costo computacional alto, es por eso que este solo debe ser utilizado para la producción de resultados finales de la simulación una vez que se ha afinado el comportamiento de la misma utilizado el otro modelo de visualización.

VII. Conclusiones

El modelo por medio de bases de turbulencia disminuye apreciablemente la complejidad de los métodos normalmente utilizados para el modelaje de los fenómenos turbulentos (Navier-Stokes), además permiten que el usuario del sistema tenga un mayor control sobre lo que desea aparezca en la simulación.

El modelaje por niveles de los campos de movimiento, permite manejar la fineza de la solución, al utilizar solo los niveles que se desee, sin necesidad de tener que calcular las ecuaciones para todo el sistema. También permite la utilización de diferentes métodos de modelaje de campos base, de tal manera que se pueden adaptar métodos tradicionales de modelaje (Navier-Stokes, Campos Aleatorios) eliminado las irregularidades que se introducen en las ecuaciones facilitando su solución, así como el poder combinar diferentes métodos.
El sistema de modelaje de tiempo por medio de procesos estocasticos permite manejar el tiempo de manera mas activa, logrando obtener mejores resultados en la simulación ya que se maneja el tiempo de manera mas adaptativa permitiendo obtener resultados mas suaves en la simulación. Este sistema de modelaje se ve reflejado en el modelo computacional del sistema en donde se manejan los procesos por medio de un anillo controlado por el tiempo, lo que además facilita la distribución de procesos.

El sistema actualmente se encuentra en proceso de implementación utilizando un esquema cliente-servidor sobre un computador Cray y computadores Alpha, separando los procesos de modelaje y calculo de la simulación (los cuales se están implementando en el Cray aprovechando la capacidad de procesamiento en paralelo), y el proceso de visualización (implementado en los Alpha).

Bibliografía

[EBE94] David S. Ebert, F. Kenton Musgrave, Dan Peachey, Ken Perlin, Steve Worley, "Texturing and Modeling a procedural approach", Academic Press Inc., Cambridge MA, 1994.

[AVE89] Law Averill M., David Kelton, Simulation Modeling and Analysis, 1989.

[FOL93] J.Foley , A. vanDam, S. Feiner, J. Hughes, Computer Graphics Principle and Practice, USA, Addison Wesley, 1992.

[STA93] Jos Stam, Eugene Fiume, Turbulent Wind Fields for Gaseous Phenomena, ACM Computer Graphics (SIGGRAPH'93), August 1993.

[KAS90] M. Kass and G.Miller. "Rapid, Stable Fluid Dynamics for Computer Graphics", ACM Computer Graphics (SIGGRAPH'90), August 1990.

[ROG84] R.S. Rogallo and P.Moin., Numerical Simulation of Turbulent Flows, Annual Review of Fluid Mechanics, 1984.

[CFD95] Computed Fluid Dynamics Journal, IEEE Press, January-December 1994, January April,1995.

[REE85] Reeves, W.T., and R. Blau, Aproximate and Probabilistic Algortihms for Shading and Rendering Particle Systems, ACM Computer Graphics (SIGGRAPH'85).

[WIT87] Witkin, A., K. Fleischer, and A. Barr, Energy Contraints on Parametrized Models, ACM Computer Graphics (SIGGRAPH'87).

[HAR86] Hartnell Tim, Replicating Reality: Exploring Computer Simulations, Ed. Anaya Mutimedia, 1986.

[STR95] Streeter Victor L., Mecanica de los Fluidos, Colombia, McGraw Hill, Octava Edición, 1995.

[BAR951] Barrero Daniel, Metodologia de Modelaje Multinivel para Fenomenos de Turbulencia. UniAndes, 1995.

[BAR952] Barrero Daniel, Modelaje y Simulación de Fenomenos Turbulentos - Bases de Turbulencia, UniAndes, 1995.

[APO86]. Apostol, Tom M., Calculus volumen 2, España, Editorial Reverté S.A., 1986.

Marco Formal para la Especificación de Implementaciones de la FFT sobre Transputers

Carlos Barra Peñaloza
Armada de Chile
Valparaíso, Chile
cbarra@inf.utfsm.cl

Horst von Brand
Universidad Técnica Federico Santa María
Valparaíso, Chile
vonbrand@inf.utfsm.cl

RESUMEN

Se presenta un marco de especificación formal, basado en TLA$^+$, para especificar topologías de redes de transputers que implementen el algoritmo de la Transformada Rápida de Fourier (FFT) En este marco se especifican el algoritmo y la arquitectura de los transputers, aislando los aspectos que definen una topología particular. Dichos aspectos son considerados en forma abstracta, lo que permite especificar una implementación dada mediante un módulo de TLA$^+$ independiente. Este trabajo se basa en [BAR96].

Introducción

La vasta investigación que se ha realizado sobre la computación paralela de la FFT incluye el desarrollo de redes de interconexión particularmente adecuadas para el intercambio de datos requerido por el algoritmo; mediciones de rendimiento de distintos sistemas paralelos (memoria compartida, memoria privada con red de interconexión, vectores, etc.); y el estudio del impacto del costo de comunicación sobre el rendimiento global, para diferentes estrategias de interconexión (ver referencias citadas en [MAV95], [FRA94], y [QUI93]). Actualmente, un aspecto importante de dicha investigación, es la explotación de los últimos avances en el diseño y fabricación de circuitos VLSI ("Very Large System Integration"), pero también existe un considerable interés en el uso de multicomputadores MIMD ("Multiple Instructions Multiple Data") donde la DFT puede ser integrada con otros algoritmos de procesamiento.

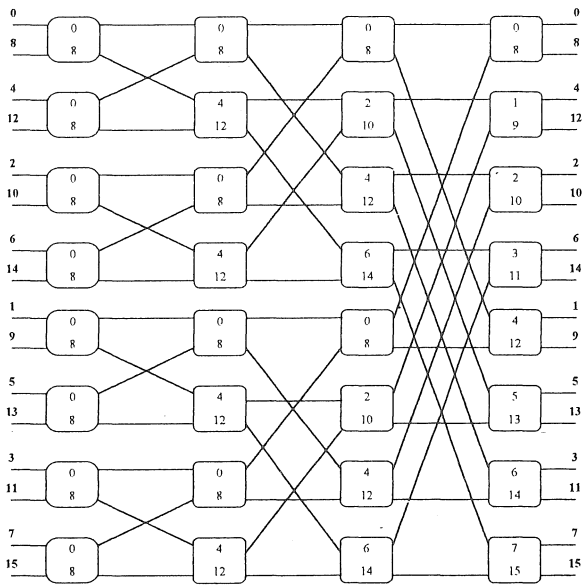
1. La Transformada Rápida de Fourier (FFT)

La transformada discreta de Fourier (DFT –"Discrete Fourier Transform") es una herramienta muy utilizada en el procesamiento de señal, en que el algoritmo comúnmente

usado para su computación es la FFT ([SDS75]). A pesar de la drástica reducción en el número de operaciones de punto flotante que se ha logrado mediante técnicas de transformación, la FFT continúa siendo una tarea intensiva en operaciones. Existen muchas aplicaciones donde los datos a ser transformados son producidos a velocidades muy altas (sistemas de tiempo real. Por otra parte, la FFT no sólo es un algoritmo intrínsecamente paralelo, si no que además su computación es completamente independiente de los datos.. La DFT transforma un vector de N componentes reales,. en un vector de N componentes complejos, mediante la multiplicación del primero por una matriz de N por N de las potencias de la N-ésima raíz primitiva de la unidad ([SBA83]). Un algoritmo secuencial, mediante el cual la FFT realiza el cálculo de la DFT, se basa en la *descomposición temporal* de los elementos del vector de entrada, lo que significa que éstos son reordenados de acuerdo a la *inversión de bits* (*bit reversal*) ([AHU74], [SDS75]), y en la estrategia de dividir y conquistar para la evaluación en los diferentes niveles de descomposición. Este algoritmo puede ser visualizado explícitamente mediante un diagrama de flujo de señal como el que se muestra en la Figura 1.1, donde se ha considerado el caso específico para $N = 16$, lo que implica 4 niveles de descomposición ($\log_2 N$).

Figura 1.1:

Diagrama de flujo de señal para N = 16



El diagrama de flujo de señal de la Figura 1.1 no permite incorporar formalmente aspectos importantes necesarios para un análisis de rendimiento, que son la arquitectura de la plataforma donde será implementado el algoritmo, la granularidad del paralelismo de la implementación y la topología de la arquitectura. Estos aspectos no son independientes entre sí y tampoco son únicos, por lo que se requiere de una especificación formal que permita introducir modificaciones y así poder realizar el análisis comparativo de diversas combinaciones de estos factores.

TLA^+ ([LLM95]) es un lenguaje formal basado en la lógica temporal de acciones (TLA –"Temporal Logic of Actions", [LLA94]) que permite especificar mediante componentes denominados módulos, que normalmente se utilizan para representar la descomposición de un sistema; por lo cual, la incorporación de las especificaciones de los componentes arquitectura, paralelismo y topología, se realizará mediante dicho lenguaje.

2. Especificación de una Arquitectura de Multicomputadores

Un transputer ([INM89]) es un componente VLSI con procesador, memoria y enlaces (*links*) de comunicaciones para conectarse directamente con otro, usando un protocolo tipo *rendezvous*. Se puede suponer que ([MAV95]) las comunicaciones a través de cada *link* ocurren concurrentemente con las de otros *links* y con la actividad de CPU ([INM89]).

3. Especificación de Aspectos Lógicos de la Arquitectura

El protocolo de comunicaciones se establece entre dos procesos, lo que plantea la necesidad de identificar los procesos tanto en forma individual, como en pares de procesos que se comunican. Esto lleva a considerar dos aspectos para la especificación de la arquitectura: es necesario implementar identificadores para los procesos (*Butterflies*), y especificar la restricción que impone el algoritmo para procesos que se comunican. Así, en la especificación del protocolo se tendrá, implícitamente, la especificación de los canales *virtuales*

de comunicación (que comunican procesos), que se diferencian de los canales reales en que estos últimos comunican procesadores. Dependiendo de la granularidad del paralelismo, cada procesador tendrá asignado un cierto número de procesos, cada uno de los cuales dispondrá de canales virtuales de comunicación implementados por canales reales, por lo que la topología involucra el mapeo de los procesos a los procesadores y la implementación de los canales virtuales.

Se visualizan tres módulos de especificación que se denominarán: *SetMap*, para especificar la identificación individual de procesos y procesadores y para definir la función de mapeo requerida por la especificación de la topología; *ProcessComm*, para especificar los pares de procesos que se comunican; y *Rendezvous*, para especificar el protocolo en base a los dos módulos anteriores. Nótese que en esta etapa no se pretende especificar la topología, sólo se especificarán las funciones necesarias para ello.

El protocolo para la comunicación entre un par de procesos debe contar con una variable que permita sincronización y una variable que represente el mensaje, las cuales deben ser del tipo correcto. La Figura 3.1 especifica el módulo *Types* que declara dicha correctitud. En la Figura 3.2 se presenta el módulo *SetMap*, que especifica cada procesador como un conjunto de procesos asignados mediante la función *Map* (que debe ser implementada como parte de la especificación de la topología). Así, cada proceso queda formalmente identificado por ⟨*i, level*⟩ y cada procesador por ⟨*j, stage*⟩, siendo *Map* una función que mapea un par identificador de proceso a un par identificador de procesador (el que ejecuta el proceso cuyo par identificador es mapeado).La Figura 3.3 muestra el módulo *ProcessComm*, que especifica un proceso como un par de variables ⟨*ctrl, msg*⟩, y que el conjunto *InterComm* está formado por los pares de procesos que se comunican en el algoritmo, cuyos pares identificadores están dados en base a las funciones *CommDown* y *CommUp* especificadas.

Figura 3.1:
Módulo de Especificación de Tipos

Module *Types*

definitions LegalValue ≡ COMPLEX
Control ≡ {"rtgd", "rtg", "rtr", "rtw"}

La Figura 3.4 muestra el módulo *Rendezvous* que especifica dos operaciones disponibles para los procesos (el ambiente de este componente): *Read* y *Write*; estas operaciones tienen dos parámetros: el canal virtual (*cp*) y el valor (*val*) que se almacena o se obtiene, respectivamente, de una variable buffer que contiene cada proceso (*msg*). Además, cada operación cambia el estado del proceso invocador (*ctrl* pasa "rtr" o "rtw") deteniendo las acciones de éste en espera de que se produzca el paso del mensaje (acción *MsgPass*). La propiedad de *fairness* para la acción de la acción *MsgPass*, asegura que ésta ocurrirá eventualmente. Así, el protocolo queda especificado por $Comm_{CH}$, y la arquitectura por *Arq*. Nótese que, de acuerdo a la arquitectura de los transputers ([INM89]), la especificación del protocolo no evita situaciones de bloqueo tales como cuando un proceso ejecuta la operación Read o Write sin que el proceso al cual va dirigido dicha operación, ejecute la operación Write o Read, respectivamente, hacia el primero. Por tanto, depende de la especificación del sistema asegurar la condición WF de la acción *MsgPass*.

Figura 3.2:
Módulo de Especificación de Identificación de procesos y procesadores

Module *SetMap*

import *Naturals*
parameters *I*, *LEVEL*, *J*, *STAGE* : constant
i, *level*, *j*, *stage* : variable

definitions $Element(M, N) \equiv \{\langle i, j\rangle : (0 \le i \in \text{Nat} < M) \wedge (0 \le j \in \text{Nat} < N)\}$
$Matrix(Set) \equiv \text{union}\{[Element(M, N) \to Set] : M, N \in \text{Nat}\}$
$Mapping \equiv [Element(I, LEVEL) \to Element(J, STAGE)]$
$t[j, stage] \equiv \{ p[i, level] : \wedge p \in \text{Matrix}(P)$
$\wedge \langle i, level\rangle \in \text{domain}(p)$
$\wedge t \in \text{Matrix}(T)$
$\wedge \langle j, stage\rangle \in \text{domain}(t)$
$\wedge Map \in Mapping$
$\wedge \langle j, stage\rangle = Map[i, level] \}$

Figura 3.3: *Módulo de especificación de proceso e intercomunicaciones*

Module *ProcessComm*

import *SetMap, Types*

definitions

$$P \equiv \left\{ \langle p[i, level].ctrl, p[i, level].msg \rangle : \begin{array}{l} \wedge\ msg \in \text{LegalValue} \\ \wedge\ ctrl \in \text{Control} \end{array} \right\}$$

$$CommDown[i, level] \equiv i - \left((i \bmod 2^{level+1}) - (i \bmod 2^{level}) \right)$$

$$CommUp[i, level] \equiv CommDown[i, level] + 2^{level}$$

$$InterComm \equiv \bigcup_{\text{domain}(p)} \left\{ \begin{array}{c} \{\langle p[i, level], p[CommUp[i, level], (level + 1)] \rangle\} \\ \cup \\ \{\langle p[i, level], p[CommDown[i, level], (level + 1)] \rangle\} \end{array} \right\}$$

Figura 3.4: *Módulo de especificación del protocolo rendezvous*

Module *Rendezvous*

import *SetMap, ProcessComm, Types*

parameter *val* : variable

assumptions $Asssump \equiv \wedge\ val \in \text{LegalValue}$
$\wedge\ \exists k \in \{0,1\} \wedge \exists \langle m,n \rangle \in \text{domain}(p) : cp[k] = p[m,n]$

environment definition

$$Init_{Ch}(cp) \equiv \forall k \in \{0,1\} : cp[k].ctrl = \text{"rtg"}$$

$$Read(cp, val) \equiv \exists k \in \{0,1\} : \begin{array}{l} \wedge\ cp[k].ctrl = \text{"rtg"} \\ \wedge\ cp[k].ctrl' = \text{"rtr"} \end{array}$$

$$Write(cp, val) \equiv \exists k \in \{0,1\} : \begin{array}{l} \wedge\ cp[k].ctrl = \text{"rtg"} \\ \wedge\ cp[k].ctrl' = \text{"rtw"} \\ \wedge\ cp[k].msg' = val \end{array}$$

$$Comm_E(cp, val) \equiv Read(cp, val) \vee Write(cp, val)$$

component definition

$$MsgPass(cp, val) \equiv \exists k \in \{0,1\} : \begin{array}{l} \wedge\ cp[k].ctrl = \text{"rtr"} \\ \wedge\ cp[1-k].ctrl = \text{"rtw"} \\ \wedge\ val' = cp[1-k].msg \\ \wedge\ cp[0].ctrl = \text{"rtg"} \\ \wedge\ cp[1].ctrl = \text{"rtg"} \\ \wedge\ \forall k : cp[k].msg' = \varnothing \end{array}$$

protocol definition

$$Comm_{Ch}(cp, val) \equiv \begin{array}{l} \wedge\ Init_{Ch}(cp, val) \\ \wedge\ \Box \left[\begin{array}{l} \vee\ Comm_E(cp, val) \\ \vee\ MsgPass(cp, val) \end{array} \right]_{\langle cp, val \rangle} \\ \wedge\ WF_{\langle cp, val \rangle}(MsgPass(cp, val)) \end{array}$$

$$Arq \equiv \bigwedge_{cp,\ val} Comm_{Ch}(cp, val)$$

4. Implementación del Sistema AA (Algoritmo-Arquitectura)

Se considerará un módulo *StateFunctions* para especificar las funciones de estado de la primera etapa y un módulo *FFT-Actions* para especificar las acciones del comportamiento de los procesos. Por otra parte, se especificará la integración de la arquitectura, pero, en realidad, la arquitectura se incorpora a la implementación a través de una topología, es decir, el protocolo de comunicaciones se manifiesta a través de la interacción de procesos, la que se produce sobre una red de conexiones físicas (topología de conexión –canales reales–) entre los procesadores a los cuales se les han asignado dichos procesos (topología de mapeo –asignación de procesos–), todo lo cual es lo que se considera como topología. Como no se pretende especificar una topología particular, se considerará una "topología virtual" sobre la cual los procesos se comunican a través de canales virtuales, bajo el protocolo de la arquitectura ya especificada. Entonces, un módulo *Virtual-Topology* contendrá la especificación de las acciones virtuales que esta topología virtual impone al comportamiento del sistema, y un módulo *SpecFFT-Virtual* contendrá la especificación del sistema algoritmo-arquitectura (AA).

La Figura 4.1. muestra el módulo *StateFunctions*, que especifica las funciones de estado necesarias para las acciones del algoritmo y para la identificación de procesos. *DatIn.Map* y *DatOut.Map* son funciones cuyos resultados son un conjunto de datos que se almacenan en un arreglo; *w.Map* es la función cuyos resultados son un conjunto de datos almacenados en una estructura tipo arreglo o tabla de valores; y *BitReversal* y *PowRoot* tienen la interpretación definida en el módulo *StateFunctions*. *F* es un conjunto de funciones que permite formalizar la notación tipo X [*i, level*] utilizada en la especificación de las acciones del algoritmo, como se muestra en la especificación del módulo *FFT-Actions* en la Figura 4.2. Los resultados se envían de un nivel a otro a través de un paso de mensaje realizado bajo el protocolo especificado por la arquitectura; lo cual se especifica en el módulo *Virtual-Topology* de la Figura 4.3. Esta topología virtual especifica que los procesos se comunican mediante *Rendezvous* para el paso de resultados, de modo que la combinación de los módulos *Virtual-Topology* y *FFT-*

Actions, representa todas las acciones del sistema, el cual se especifica en módulo *SpecFFT-Virtual* de la Figura 4.4. Esta especificación implementa al algoritmo considerando las restricciones que impone la arquitectura, por lo que queda especificado formalmente el sistema AA, representado por la fórmula *FFT-Virtual*; además, la operación *Butterfly* especifica la granularidad básica que se ha considerado y que deberá ser mantenida en la especificación de la topología de mapeo (para otras granularidades, habrá que modificar esta especificación y las siguientes). Así, en este nivel de abstracción, se tiene la especificación del sistema incluyendo algoritmo, arquitectura y topología virtual; esta última, considera algún tipo de mapeo procesos-procesadores y algún tipo de red de conexión entre procesadores, que la implementan de modo que *FFT-Virtual* es válida.

Figura 4.1: *Módulo de especificación de funciones de estado*

Module *StateFunctions*

import *Naturals*

parameters N : constant

F, f, m, n : variables

assumption $ParamAssump \equiv M, m, n \in \text{Nat}$

definitions $BitReversal[m] \equiv \sum_{k=0}^{(log_2 N)-1} \left(2^m\right) \cdot \frac{m \bmod 2^{((log_2 N)-k)} - m \bmod 2^{((log_2 N)-(k+1))}}{2^{((log_2 N)-(k+1))}}$

include *SetMap* **as** *DatIn* **with** $I \leftarrow N, LEVEL \leftarrow 1$:

$\forall m \in \text{domain}\left(DatIn.Map\right) : DatIn.Map[m] \in \text{LegalValue}$

include *SetMap* **as** *DatOut* **with** $I \leftarrow N, LEVEL \leftarrow 1$:

$\forall m \in \text{domain}\left(DatOut.Map\right) : DatOut.Map[m] \in \text{LegalValue}$

include *SetMap* **as** w **with** $I \leftarrow N/2 - 1, LEVEL \leftarrow 1$:

$\forall m \in \text{domain}\left(w.Map\right) : w.Map[m] = e^{-(\sqrt{-1})2\pi n/N}$

$PowRoot[m,n] \equiv w.Map\left[N \cdot 2^{-(m+1)} \cdot m \bmod 2^n\right]$

$F[m,n] \equiv \left[\langle m,n\rangle \rightarrow p[m,n].f\right]$

Figura 4.2: *Módulo de especificación de acciones*

Module *FFT–Actions*

import *Types, StateFunctions, ProcessComm, Rendezvous*

parameters x, y : variables

$result$: boolcan

assumption $ParamAssump \equiv$
$\wedge\ x, y \in \text{LegalValue}$
$\wedge\ cp \in InterComm : cp[0] = p[i, level]$
$\wedge\ Ctrl \in F : Ctrl[i, level] = p[i, level].ctrl$
$\wedge\ Result \in F : Result[i, level] = p[i, level].result$
$\wedge\ X \in F : X[i, level] = p[i, level].x$
$\wedge\ Y \in F : Y[i, level] = p[i, level].y$

definitions

$$Init(i, level) \equiv \begin{pmatrix} \wedge\ Ctrl[i, level] = \text{"rtg"} \wedge Ctrl[i, 0] = \text{"rtgd"} \\ \wedge\ Result[i, level] = \text{False} \\ \wedge\ \text{if}\,(CommDown[i, level - 1] = i)\ \text{then}\ (r = 1)\ \text{else}\ (r = -1) \end{pmatrix}$$

$$GetDat(i, level) \equiv \begin{pmatrix} \wedge\ Ctrl[i, level] = \text{"rtgd"} \wedge Result[i, level] = \text{False} \\ \wedge\ X'[i, level] = DatIn.Map[BitReversal[i]] \\ \wedge\ Y'[i, level] = DatIn.Map[BitReversal[i] + N/2] \\ \wedge\ Ctrl'[i, level] = \text{"rtg"} \end{pmatrix}$$

$$CalcFFT(i, level) \equiv \begin{pmatrix} \wedge\ Ctrl[i, level] = \text{"rtg"} \wedge Result[i, level] = \text{False} \\ \wedge\ X'[i, level] = X[i, level] + Y[i, level] \cdot PowRoot[i, level] \\ \wedge\ Y'[i, level] = X[i, level] - Y[i, level] \cdot PowRoot[i, level] \\ \wedge\ \text{if}\,(level = 0) \\ \quad \text{then}\ Ctrl'[i, level] = \text{"rtgd"} \\ \wedge\ Result'[i, level] = \text{True} \end{pmatrix}$$

$$PutDat(i, level) \equiv \begin{pmatrix} \wedge\ (Result[i, level] = \text{True} \wedge level = log_2 N) \\ \wedge\ DatOut.Map'[BitReversal[i]] = X'[i, level] \\ \wedge\ DatOut.Map'[BitReversal[i] + N/2] = Y'[i, level] \\ \wedge\ Result'[i, level] = \text{False} \end{pmatrix}$$

Figura 4.3: *Módulo de especificación de la topología virtual*

Module *Virtual - Topology*

import *FFT - Actions, ProcessComm*

definitions

$$ReceiveResult(i, level) \equiv \left(\begin{array}{l} \\ \wedge Ctrl[i, level] = \text{"rtg"} \\ \wedge Read(cp, X[i, level]) : \\ \quad cp[1] = p[CommDown[i, level-1], level-1] \\ \wedge Read(cp, Y[i, level]) : \\ \quad cp[1] = p[CommUp[i, level-1], level-1] \end{array} \right)$$

$$SendResult(i, level) \equiv \left(\begin{array}{l} \wedge Result[i, level] = \text{True} \\ \wedge Write(cp, X[i, level]) : \\ \quad cp[1] = p[CommDown[i, level], level+1] \\ \wedge Write(cp, Y[i, level]) : \\ \quad cp[1] = p[CommUp[i, level], level+1] \\ \wedge \text{if}(level \neq log_2 N) \\ \quad \text{then } Result'[i, level] = \text{False} \end{array} \right)$$

Figura 4.4: *Módulo de especificación de sistema AA*

Module *SpecFFT–Virtual*

import *FFT–Actions, Virtual-Topology*

definitions

$$Butterfly(i, level) \equiv \vee GetDat(i, level) \vee CalcFFT(i, level) \vee ReceiveResult(i, level) \vee PutDat(i, level) \vee SendResult(i, level)$$

$$hvar[i, level] \equiv \langle Ctrl(i, level), Msg(i, level), cp, Result(i, level) \rangle$$

$$vvar[i, level] \equiv \langle X(i, level), Y(i, level) \rangle$$

$$var[i, level] \equiv \langle hvar[i, level], vvar[i, level] \rangle$$

$$NodeFFT(i, level) \equiv Init(i, level) \wedge \Box[Butterfly(i, level)]_{var[i, level]}$$

$$FFT–Virtual \equiv \bigwedge_{i, level} (\exists hvar[i, level] : NodeFFT(i, level))$$

5. Especificación de Aspectos Físicos

Para especificar la red de conexiones físicas entre los transputers y el sincronismo de comunicaciones de una topología particular, se requiere la especificación de la figura 5.1, donde *Wire* es una función que permite una sola conexión abstracta de un *link* con otro de otro transputer; nótese que se incluye la posibilidad de que un transputer se comunique consigo mismo (el caso de dos procesos en un mismo procesador que se comunican). Un canal real de comunicación es aquel que existe entre dos procesos que se comunican, cuando ambos están asignados a procesadores (transputers) entre los que se cumple la función *Wire*. Así, la red de conexión entre transputers queda especificada como el conjunto *Channel* de pares de procesos *cp* que cumplen con la restricción anterior.

Figura 5.1: *Módulo de especificación de conectividad*

Module *Connectivity*

import *SetMap, ProcessComm*

parameters $j^*, stage^*, j^{**}, stage^{**}, cp$: variable

assumption *Assump* $\equiv$
$\wedge\ cp \in InterComm$
$\wedge\ \langle j^*, stage^* \rangle, \langle j^{**}, stage^{**} \rangle \in \text{domain}\ (t)$
$\wedge\ \text{LINK} \equiv \{0, 1, 2, 3\}$
$\wedge\ link \in \text{LINK}$

definitions

$$TNet \equiv [\langle \text{domain}\ (t) \times LINK \rangle \rightarrow \langle \text{domain}\ (t) \times LINK \rangle]$$

$$Wire\ [j, stage, link] \equiv \langle j^*, stage^*, link^* \rangle$$
$$: \wedge\ Wire \in TNet$$
$$\wedge\ Wire\ [j^*, stage^*, link^*] = \langle j, stage, link \rangle$$
$$\wedge\ Wire\ [j, stage, link] = Wire\ [j^{**}, stage^{**}, link^{**}]$$
$$\Rightarrow \langle j, stage, link \rangle = \langle j^*, stage^*, link^* \rangle$$

export

$$Connection\ [j, stage] \equiv \left\{ \begin{array}{l} \langle j^*, stage^* \rangle : \\ \left(\begin{array}{l} \exists\ link, link^* : \\ \langle j^*, stage^*, link^* \rangle = Wire\ [j, stage, link] \end{array} \right) \end{array} \right\}$$

$$Channel \equiv \left\{ \begin{array}{l} cp : \\ \left(\begin{array}{l} \exists\ k \in \{0,1\}, \langle j^*, stage^* \rangle \in Connection\ [j, stage] : \\ \left(\wedge\ cp\ [k] \in t\ [j, stage] \wedge\ cp\ [1-k] \in t\ [j^*, stage^*] \right) \end{array} \right) \end{array} \right\}$$

Los aspectos referidos (conexión y sincronismo de comunicaciones), se pueden resolver mediante la especificación del conjunto *Connection* y la especificación adecuada de las acciones *ReceiveResult* y *SendResult*, de modo que las operaciones *Read* y *Write* se ejecuten sobre *cp's* (canales virtuales definidos por pares de procesos que se comunican) del conjunto *Channel*. En una próxima etapa de especificación, se deberán especificar *Map*, *Connectivity*, *ReceiveResult* y *SendResult*, de acuerdo a la topología que se requiera; esto significa especificar cómo se asignan los procesos a los transputers, cómo estos transputers serán físicamente conectados y cómo los procesos se comunicarán a través de canales establecidos por dichas conexiones. Si se considera que estas especificaciones estarán contenidas en un módulo, entonces es posible hacer una especificación general del sistema AAT, importando dicho módulo.

6. Especificación del Sistema AAT (Algoritmo-Arquitectura-Topología)

Dado que se cuenta con la especificación de las acciones del sistema (módulo *FFT-Actions*) y la especificación de la topología (cualquiera sea), entonces, al igual que se especificó el sistema AA, se especifica el sistema AAT como se muestra en la Figura 6.1.

Figura 6.1: *Módulo de especificación del sistema AAT*

Module *SpecFFT*

import $FFT-Actions, Any-Topology$

definitions

$$Butterfly\ (i, level) \equiv \vee\ GetDat\ (i, level) \vee CalcFFT\ (i, level) \vee Any.ReceiveResult\ (i, level) \vee PutDat\ (i, level) \vee Any.SendResult\ (i, level)$$

$$hvar\ [i, level] \equiv \langle Ctrl\ (i, level), Msg\ (i, level), cp, Result\ (i, level) \rangle$$

$$vvar\ [i, level] \equiv \langle X\ (i, level), Y\ (i, level) \rangle$$

$$var\ [i, level] \equiv \langle hvar\ [i, level], vvar\ [i, level] \rangle$$

$$NodeFFT\ (i, level) \equiv Init\ (i, level) \wedge \Box\ [Butterfly\ (i, level)]_{var\ [i, level]}$$

$$FFT \equiv \bigwedge_{i, level} (\exists\ hvar\ [i, level] : NodeFFT\ (i, level))$$

La especificación del sistema AAT queda dada por la fórmula *FFT* d módulo *SpecFFT*, donde sólo basta reemplazar el nombre del módulo importado *Any-Topolog* por aquel del módulo de especificación de alguna topología particular. Por lo tanto, *FF* implementa a Φ si la especificación de la topología es válida, lo cual puede ser demostrad probando la validez de la fórmula $FFT \Rightarrow \Phi$, mediante los axiomas y reglas de demostración d TLA ([LLA94]), lo que se deja para un trabajo futuro por no corresponder al objetivo de este además, del mismo modo, se pueden probar las fórmulas *FFT-Virtual* $\Rightarrow \Phi$ y $FFT \Rightarrow$ *FFT-Virtual*. Así, sólo resta hacer una especificación válida de la(s) topología(s) en base a lo planteado en la sección anterior, para obtener la(s) especificación(es) completa(s) de el(los) sistema(s) AAT('s), para realizar un estudio de rendimiento.

Bibliografía

[AHU74] Aho A. V., J. E. Hopcroft, J. D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley, Reading Mas., 1974.

[BAR96] Carlos Barra P,"Métodos formales en el Análisis y Diseño de Algoritmos Paralelos sobre Arquitecturas de Multicomputadores: un caso práctico ", Tesis de Magister en Ingeniería Informática, Universidad Técnica Federico Santa María, Valparaíso, Chile, 1996.

[FRA94] Paraskevi Fragopoulou, Selim G. Akl, "A Parallel Algorithm for Computing Fourier Transforms on the Star Graph", IEEE Transaction on Software Engineering, vol. 5, N° 5, May 1994.

[INM89] Inmos Ltd., "The Transputer Databook", Consolidated Printers, Berkeley, USA, 1989.

[LLA94] Leslie Lamport, "The Temporal Logic of Actions", ACM Transactions on Programming Languages and Systems, May 1994.

[LLM95] Leslie Lamport, "TLA$^+$", http://www.research.digital.com/SRC/personal/ Leslie_Lamport /tla/tla.html, 23 Mar 1995.

[MAV95] Antonio Mazzeo and Umberto Villano, "Parallel 1D-FFT Computation on Constant-valence Multicomputers", Software-Practice and Experience, vol. 25, June 1995.

[QUI93] Michael Quin, "Parallel Computing", McGraw-Hill Book Company, 1993.

[SDS75] Samuel D. Stearns, "Digital Signal Analysis", Hyden Books Inc., 1975.

[SBA83] Sara Baase, "Computer Algorithms: Introduction to Design and Analysis", Addison-Wesley, Reading Mas., 1983.

Partición de Sistemas de Ecuaciones para su Resolución Distribuida[1]

Benjamín Barán[+] *, Diana Benítez* y Rodrigo Ramos[+]
[+]Centro Nacional de Computación y Facultad de Ingeniería, Universidad Nacional de Asunción
Fax: (595) (21) 585619
*Facultad de Ciencias y Tecnología, Universidad Católica Nuestra Señora de la Asunción
Fax: (595) (21) 310587
E-mail: bbaran@una.py

Resumen

Este trabajo propone un método para descomponer sistemas de ecuaciones, de manera a facilitar su resolución utilizando ambientes distribuidos heterogéneos como computadores paralelos y redes de computadores. La presente propuesta asigna a los procesadores de un sistema distribuido heterogéneo distintos números de ecuaciones según sea la performance relativa de los procesadores a ser utilizados, de forma tal que su aplicación sea válida para un sistema general de ecuaciones. Asímismo, el método presentado permite utilizar la técnica de solapamiento parcial (*partial overlapping*) para mejorar la convergencia. Además, se introduce un parámetro de selección para escoger la mejor partición de un conjunto dado. Se presentan resultados experimentales que avalan la presente propuesta.

Palabras Claves: Computación distribuida, convergencia de métodos iterativos, performance relativa.

[1]Este trabajo fue iniciado en el marco del proyecto UNESCO - ROSTLAC 885.610-4; con el auspicio de la Universidad Nacional de Asunción (UNA) y la Universidad Católica Nuestra Señora de la Asunción (UCA) y contó con el patrocinio de la Dirección de Investigaciones Postgrado y Relaciones Internacionales (DIPRI) de la UNA.

1. INTRODUCCIÓN

Las implementaciones paralelas de algoritmos iterativos son una opción importante dentro del contexto de la computación distribuida [1]. Para aprovechar sus ventajas en la resolución de sistemas de ecuaciones de gran porte, es necesario descomponer el problema dado en subproblemas menores, los cuales pueden ser asignados a los distintos procesadores del sistema distribuido para su resolución paralela. Esta descomposición está condicionada por dos factores:

a) la capacidad de procesamiento relativo de cada máquina del sistema distribuido, factor que determina las dimensiones de los subproblemas asignados a cada procesador;

b) el grado de acoplamiento que puedan tener los subproblemas entre sí, lo que determina la dependencia entre las variables y por consiguiente, influye en la convergencia del algoritmo [2].

Así, el sistema debe descomponerse de forma tal que a cada procesador se le asigne un subproblema de dimensión proporcional a su performance, y que la dependencia entre las variables actualizadas por cada procesador facilite la convergencia del algoritmo iterativo a ser utilizado.

Desde la década del 60 [3] fueron publicados diversos métodos de descomposición, en su mayoría diseñados para procesadores, problemas o algoritmos bien específicos [4-5]. Así, Vale et al. [6] formuló una propuesta para sistemas de ecuaciones lineales y simétricos utilizando computadores paralelos. Sin embargo, la *Descomposición* ε [7] que desprecia conexiones débiles, es la más utilizada en la actualidad.

Inspirada en el trabajo de Vale et al. [6], se desarrolla la presente propuesta que permite generar particiones en número y tamaño controlables por el operador, válidas para un sistema general de ecuaciones. Además, se introduce un parámetro que permite elegirr una buena partición de un conjunto de particiones y se sugiere la utilización de solapamiento parcial (*partial overlapping*) [8].

La Sección 2 formula matemáticamente el problema, mientras que la Sección 3 presenta el método propuesto. Resultados experimentales se presentan en la Sección 4 y las conclusiones en la Sección 5.

2. FORMULACIÓN DEL PROBLEMA

Considerando un sistema de *n* ecuaciones algebraicas con n incógnitas dado por:

$$\Phi(x) = 0, \qquad x \in \Re^n, \qquad \Phi(\cdot): \Re^n \rightarrow \Re^n \tag{2.1}$$

definido en un dominio $D \subset \Re^n$; se desea resolver (2.1) utilizando un sistema distribuido con p procesadores, $n \geq p$. Para esto, se debe descomponer el problema de forma tal que cada procesador *resuelva* solamente una parte del sistema, comunicando sus resultados parciales a los otros procesadores.

Sea la Descomposición Cartesiana: $\Re^n = \Re^{n_1} \times \cdots \times \Re^{n_p}, \quad n_1 + \cdots + n_p = n$ (2.2)

con $D \subset \Re^n$ un dominio tal que $D = D_1 \times \cdots \times D_p, \quad D_i \subset \Re^{n_i}, \quad \forall i \in \{1,\ldots,p\};$ (2.3)

entonces: $x = \left[x_1^T \quad \cdots \quad x_p^T\right]^T, \; x_i \in D_i, \; \forall\, i \in \{1,\ldots,p\}.$ (2.4)

Análogamente, $\Phi(x) = \left[\Phi_1^T(x) \quad \cdots \quad \Phi_p^T(x)\right]^T, \quad \Phi_i(\cdot): D \rightarrow \Re^{n_i}$ (2.5)

Consecuentemente, la ecuación a ser resuelta en cada procesador *i* (problema local), estará dada por:

$$\Phi_i(x) = 0, \quad \forall i \in \{1,\ldots,p\} \tag{2.6}$$

NOTA 1. *La descomposición adecuada de la función $\Phi(x)$ en p funciones $\Phi_i(x)$ es el objetivo central del presente trabajo.*

Para resolver el problema será utilizado un algoritmo iterativo de la forma:

$$x^{k+1} = G(x^k) \tag{2.7}$$

donde *k* representa la iteración. Por lo tanto, en forma análoga a (2.4) y (2.5):

$$G(x) = \left[G_1^T(x) \quad \cdots \quad G_p^T(x)\right]^T, \qquad G_i(\cdot): D \rightarrow D_i, \qquad \forall i \in \{1,\ldots,p\} \tag{2.8}$$

La implementación síncrona de (2.7) para el procesador *i* será entonces:

$$x_i^{k+1} = G_i(x^k) \tag{2.9}$$

Considerando un contexto asíncrono, se denota como $x_j(d_j^i(k))$ al valor de x_j enviado desde el

procesador *j* y disponible en el procesador *i* al iniciar su iteración *k*. Denotamos así como $x^i(k)$ al vector *x* disponible en el procesador *i* en el momento de su iteración *k*, esto es:

$$x^i(k) = \left[x_1^T(d_1^i(k)) \quad \cdots \quad x_p^T(d_p^i(k))\right]^T, \qquad \forall\, i \in \{1,\ldots,p\} \tag{2.10}$$

Usando esta notación [9], podemos escribir el algoritmo asíncrono basado en (2.7) como:

$$x_i(k+1) = G_i(x^i(k)), \qquad \forall i \in \{1,\ldots,p\} \tag{2.11}$$

En [1] se deriva una condición suficiente de convergencia de (2.11), bajo las siguientes hipótesis:

Hipótesis 1 (Unicidad de la solución): *dado el sistema de ecuaciones (2.1) que puede ser reescrito como (2.6), definido en un conjunto cerrado* $D \subset \Re^n$ *que satisface (2.3), existe un operador* $G(\cdot)$ *de la forma (2.8), tal que* $G(D) \subset D$ *y adicionalmente D presenta un y solamente un punto fijo*

$$x^* = \left[\left(x_1^*\right)^T \quad \cdots \quad \left(x_p^*\right)^T\right]^T, \qquad x_i^* \in D_i, \qquad \forall\, i \in \{1,\ldots,p\}$$

del operador G. El punto fijo x^* *es a su vez solución de (2.1).*

Hipótesis 2 (Asincronismo parcial): $\exists\, d \in \mathrm{N}$, $\forall k \in \mathrm{N}$, *tal que* $d_j^i(k) \in \{k,\ldots,k-d\}$ *donde* $(k-d)$ *es el máximo atraso posible, determinado por la medida de asincronismo d.*

Hipótesis 3 (Bloque-Lipschitz): *cada operador* $G_i(x)$ *en (2.11) es Bloque-Lipschitz en D, esto es:*

$$\left\| G_i(x) - G_i(y) \right\| \le \sum l_{ij} \left\| x_i - y_j \right\|, \qquad \forall x, y \in D$$

Se demuestra así el siguiente teorema [1]:

Teorema 1 (Condición suficiente de convergencia)

Bajo las hipótesis 1 a 3, el algoritmo asíncrono representado por (2.11) converge a un único punto fijo x^* *en D si* $\rho(H) < 1$, *donde* ρ *representa el radio espectral y la matriz de comparación H está dada por:*

$$H = (h_{ij}) \in \Re^{p\,x\,p}, \qquad con\ h_{ij} = l_{ij}$$

En resumen, el radio espectral de la matriz de comparación es un parámetro que nos permitiría asegurar a

priori que el algoritmo converge, y por consiguiente podría ser utilizado para escoger buenas particiones.

La técnica de solapamiento parcial (*partial overlapping*) fue introducida para mejorar la convergencia de los métodos de resolución en bloque, y consiste en resolver una misma ecuación repetidamente en varios bloques (o procesadores, en una implementación paralela) [8].

3. PROPUESTA DEL TRABAJO

El método de descomposición propuesto utiliza una matriz cuadrada no negativa $M = \{m_{ij}\}$ ($m_{ij} \geq 0$) de dimensión n x n y elementos diagonales no nulos ($m_{ii} > 0$), cuyos elementos m_{ij} y m_{ji} ($i \neq j$) representan el acoplamiento existente entre las variables x_i y x_j. En el caso lineal ($Ax=b$), los elementos m_{ij} de la matriz M están dados por $m_{ij} = |a_{ij}|$, donde A es la matriz de coeficientes del sistema o una versión reordenada de la misma. Si el sistema a ser resuelto fuera no lineal, la matriz M dependerá del método de resolución.

Decimos que las incógnitas x_i y x_j no son adyacentes si $m_{ij} = m_{ji} = 0$; caso contrario x_i y x_j son adyacentes. En caso de ser adyacentes, se habla de variables débilmente acopladas (si m_{ij} y m_{ji} son pequeñas) o fuertemente acopladas (si m_{ij} ó m_{ji} son muy grandes).

Básicamente, el método consiste en la formación de p subproblemas a partir de p incógnitas iniciales, denominadas *semillas*, donde p representa también el número de procesadores. A estas incógnitas semillas se van agregando las demás incógnitas hasta que el sistema quede totalmente descompuesto. La determinación de las incógnitas a ser anexadas a una dada semilla depende de un ranking de incógnitas basado en pesos previamente asignados. A cada paso del proceso, la incógnita a ser agregada al subproblema es la adyacente más pesada. En caso que dos o más subproblemas seleccionen la misma incógnita como candidata, esta se asignará al subproblema con el cual posea un mayor acoplamiento. Si no es posible desempatar y la incógnita en disputa posee un peso significativo, se puede optar por solapamiento parcial. Para esto se asigna la incógnita en disputa a todos los subproblemas que la hayan seleccionado.

A medida que los subproblemas van siendo formados, debe verificarse que el tamaño de los mismos se

encuentre en la proporción deseada (performance relativa de los procesadores: w). Para esto, se anexan incógnitas sólo a aquellos subproblemas que aun no alcanzaron dicha proporción. El proceso de agrupamiento de incógnitas continua hasta que todas las incógnitas hayan sido asignadas a algún subproblema. En caso de obtenerse más de una descomposición, se utiliza un criterio de selección para elegir la partición más conveniente.

Basados en el principio descrito, se propone el siguiente método de cuatro etapas (ver detalles en [9]):

1) **Clasificación de incógnitas (Algoritmo 1):** Se hace un ranking de pesos que indica el grado de acoplamiento de cada incógnita al sistema. El peso P_i de una incógnita x_i esta dado por

$$P_i = \sum_{j=1 \; j \neq i}^{n} (m_{ij} / m_{ii})^{z_{ij}} \quad con \quad z_{ij} = m_{ij} / (m_{ii} / D) \qquad \forall i \in \{1,...,n\} \tag{3.1}$$

donde $$D = \frac{1}{nc} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} m_{ij} \;, \qquad nc = \textit{número de } m_{ij} \textit{ diferentes de cero}$$

2) **Selección de semillas (Algoritmo 2):** Deben seleccionarse como semillas a las incógnitas que sean centro de aglutinamiento de incógnitas y que no se encuentren fuertemente acopladas unas con otras.

3) **Formación de particiones (Algoritmo 3):** Una vez determinadas las semillas para cada subsistema, es necesario adicionar las demás incógnitas a los subconjuntos adecuados.

4) **Selección de la mejor partición (Algoritmo 4):** Pueden ser determinados varios conjuntos de semillas que a su vez servirán de pie para generar diferentes particiones. Para identificar a la mejor partición de todas las obtenidas se utiliza un parámetro de selección. En [2] se proponen varios parámetros posibles de selección, recomendándose como óptimo al Parámetro de Acoplamiento "Par_A", dado por:

$$\text{Par_A} = \sum m_{ij} \;, \qquad \forall i, j \; \textit{tal que} \; x_i \; y \; x_j \; \textit{pertenecen a subproblemas diferentes}$$

En base a lo expuesto en la Sección 2, este trabajo propone como parámetro de selección a $\rho(H)$, siendo elegida como óptima aquella partición que presente el mínimo valor del parámetro de selección. La efectividad del parámetro $\rho(H)$ frente a Par_A quedará demostrada por los resultados experimentales.

Algoritmo 1: Clasificación de incógnitas

Entrada: matriz M

DESDE i = 1 hasta n /* para cada una de las p incógnitas */
  Calcular P_i según la ecuación (3.1);
  Incluir a P_i en el ranking;

Salida: pesos P_i de las n incógnitas ordenadas según un ranking de pesos

Algoritmo 2: Selección de semillas

Entradas: ternas *vlim*, *ngrup* y *nvec*, matriz M, pesos de las incógnitas P_i , nº de procesadores p.
vlim, es el peso mínimo requerido para considerar una incógnita como candidata a semilla;
ngrup, es el n° de incógnitas a agrupar alrededor de cada candidata a semilla y
nvec, es el parámetro utilizado para evitar que dos incógnitas muy acopladas sean semillas.

PARA cada terna seleccionada /* Pueden ser determinadas por el usuario o por criterios estadísticos [2]*/
  Inicializar el conjunto K como vacío; /* Conjunto de incógnitas candidatas a semillas */
  PARA cada una de las incógnitas x_i
    SI peso $P_i \geq vlim$ ENTONCES
      Incluir la incógnita x_i en K ;
  PARA cada una de las incógnitas x_i en K
    Inicializar el conjunto I_i ; /* Conjunto de incógnitas agrupadas a las candidatas a semillas */
    Incluir la incógnita x_i en I_i ;
    Inicializar el conjunto CIA_i ; /* Conjunto de Incógnitas Adyacentes al conjunto I_i */
    Incluir en CIA_i las incógnitas adyacentes a la incógnita x_i ;
    DESDE 1 hasta *ngrup* /* agrupar *ngrup* incógnitas alrededor de cada candidata */
      Mover la incógnita de mayor peso del CIA_i al I_i ;
      Incluir en CIA_i , las nuevas incógnitas adyacentes a la recientemente movida;
  PARA cada conjunto I_i
    Calcular la sumatoria de los pesos de todas sus incógnitas en I_i ;
  Inicializar el conjunto S; /* S: Conjunto de semillas de una partición */
  Seleccionar de K la incógnita x_k cuyo conjunto I posea mayor sumatoria de peso ;
  Incluir x_k en S como primera semilla ;
  Eliminar x_k de K ;
  MIENTRAS el número de semillas $\leq p$
    Seleccionar en K la incógnita x_s que posea la mayor sumatoria de pesos ;
    SI x_s no está entre las *nvec* primeras incógnitas de los conjuntos I de las semillas
    ya seleccionadas ENTONCES
      Elegir x_s como semilla ;
      Incluir x_s en S ;
      Eliminar x_s de K;
    DE_LO_CONTRARIO
      Eliminar x_s de K ;

Salida: para cada terna se tendrá un conjunto S de p semillas

Algoritmo 3: Formación de particiones

Entradas: conjunto de semillas $S = \{s_1,...,s_p\}$, matriz M, pesos de las incógnitas P_i, $LimOver$: $LimOver$: mínimo peso requerido para realizar solapamiento parcial.

Inicializar los vectores $C \in N^p$ y $Q \in N^p$; /* vectores de estado para controlar el desbalanceo según w; c_i es el nº de incógnitas en el subproblema i en un momento dado; $q_i = c_i/w_i$ */
PARA cada semilla $s_i \in S$
  Inicializar el conjunto J_i ; /* conjunto de incógnitas de un subproblema */
  Incluir en J_i la semilla s_i ;
  Inicializar un conjunto CIA_i ; /* Conjunto de Incógnitas Adyacentes */
  Incluir en CIA_i las incógnitas adyacentes a la semilla s_i ;
Actualizar los vectores C y Q ;
MIENTRAS existan incógnitas no agrupadas
  PARA todos los conjuntos J_i que necesitan anexar incógnitas
    Seleccionar como incógnita candidata a ser incluida a la de mayor peso de su CIA ;
    Ver si existen coincidencias de candidatas ;
  SI existe coincidencia en la incógnita candidata $\hat{x}_k$ ENTONCES
    Seleccionar el valor del mayor acoplamiento entre las incógnitas ya incluidas en J_i y la incógnita candidata $\hat{x}_k$ a ser incluida en J_i ;
    SI existen coincidencias de valor de mayor acoplamiento ENTONCES
      SI se tiene opción de solapamiento ENTONCES
        SI el peso de la incógnita $\hat{x}_k$ es ≥ $LimOver$ ENTONCES
          Incluir $\hat{x}_k$ en todos los subconjuntos J coincidentes ;
        DE_LO_CONTRARIO
          Incluir $\hat{x}_k$ en el primer J que pelea por $\hat{x}_k$;
      DE_LO_CONTRARIO
        Incluir $\hat{x}_k$ en el primer J que pelea por $\hat{x}_k$;
    DE_LO_CONTRARIO
      Incluir $\hat{x}_k$ en J con el que tenga mayor acoplamiento ;
  DE_LO_CONTRARIO
    Incluir $\hat{x}_k$ en el primer J_i;
  Eliminar $\hat{x}_k$ de todos los CIAs de J ;
  Incluir en los CIAs correspondientes las incógnitas adyacentes a las incluidas ;
  Actualizar C y Q ;

Salida: partición en p subproblemas.

Algoritmo 4: Selección de la mejor partición

Entrada: las diversas particiones generada por el Algoritmo 3

PARA cada partición
  Calcular $\rho(H)$;
  Incluir el valor de $\rho(H)$ en el ranking;
Elegir como mejor partición la de menor valor de $\rho(H)$;

Salidas: mejor partición generada por el método propuesto, ranking de particiones.

4. RESULTADOS EXPERIMENTALES

La eficiencia del método descrito fue verificada experimentalmente utilizando sistemas de ecuaciones lineales y no lineales. Las diferentes particiones generadas por el método propuesto fueron comparadas con otras particiones determinadas aleatoriamente y con particiones generadas por la *Descomposición* ε [7]. Además se realizó un estudio comparativo entre los parámetros de selección $\rho(H)$ y Par_A.

4.1. Sistema lineal de ecuaciones.

Se presentan resultados experimentales de un sistema lineal de 10 ecuaciones y 10 incógnitas [Ejemplo A, 10]. Se hizo una descomposición exhaustiva, resolviéndose las 45[#] particiones posibles utilizando el método bloque-Jacobi, considerando una performance relativa $w = [4\ 1]^T$. Se hicieron mediciones del número de iteraciones hasta la convergencia y se calcularon los parámetros $\rho(H)$ y Par_A.

En la Tabla 4, se observa que las particiones generadas por el método propuesto están dentro de las mejores particiones posibles y que son superiores al promedio. Como puede notarse, la partición obtenida por el método propuesto (nº 7) es mejor que la partición sugerida por la *Descomposición* ε (nº 19).

nº de partición	Iteraciones[+]	Posición en el ranking		$\rho(H)$	Par_A
		del nº de iteraciones*	de particiones[#]		
7	4,33	2º	2º	0,0551	10
23	4,83	4º	6º	0,0709	8,6
44	4,67	3º	4º	0,0794	12,7
2	4,83	4º	7º	0,1008	10,5
39	5	5º	12º	0,1014	10,6
18	6	9º	21º	0,1352	11,3
[ε]19	5,5	6º	16º	0,1183	7,9

[+] El nº de iteraciones es un promedio de pruebas realizadas con seis valores posibles de tolerancia.
* Existen 16 posibles valores del nº de iteraciones que van desde 4.17 hasta 7.33; siendo 5.78 el promedio.
[#] Existen $C_2^{10} = \frac{10!}{(10-2)!\,2!} = 45$ posibles particiones. [ε] Partición generada por la *Descomposición* ε.

Tabla 4: Resultados experimentales del Ejemplo A

Se puede observar además, que si se considera a Par_A como parámetro de selección, entonces sería elegida la partición nº 23 a pesar de ser inferior a la seleccionada por el parámetro $\rho(H)$.

En otros ejemplos de sistemas lineales [10], las particiones generadas por el método propuesto son las óptimas, al igual que las sugeridas por la *Descomposición ε*.

Finalmente, se destaca que el coeficiente de correlación entre el número de iteraciones y $\rho(H)$ fue de 0,96 frente a una correlación de solo 0,15 entre el número de iteraciones y Par_A, lo que enfatiza las bondades de $\rho(H)$ (frente a Par_A) como parámetro de selección.

4.2. Sistema no lineal de ecuaciones

Se aplicó la metodología propuesta a la descomposición del sistema IEEE de 14 barras para la resolución del problema de flujo de potencia. Dicho problema puede ser formulado como un sistema cuasi-lineal de ecuaciones [11] de la forma: $Yx = I(x)$, $Y \in C^{n \times n}$, $x \in C^n$, $I(x) \in C^n$, donde Y es la matriz de admitancias de la red, x el vector de voltajes de las barras e $I(x)$ el vector de corrientes inyectadas.

La matriz de acoplamientos M utilizada será la misma matriz Y, teniendo en cuenta que ésta representa en forma efectiva al acoplamiento entre los voltajes de las barras.

Este sistema fue resuelto en forma síncrona (2.9) y asíncrona (2.11), utilizando un sistema distribuido constituido por dos *workstations*: SUN SPARC-Station 5 y DEC 3000, con $w = [4\ 1]^T$.

La Tabla 5 muestra las posiciones de las 2 particiones generadas por el método propuesto en diversos tipos de ranking, tanto para el caso síncrono como para el asíncrono. La partición 59 es la seleccionada por el método propuesto, resultando para el caso asíncrono la partición óptima.

Número de Partición	Posición[+] en el Ranking de						$\rho(H)$	Par_A
	Iteraciones.		Tiempo Real		Tiempo de CPU			
	sinc.	asinc.	sinc.	asinc.	sinc.	asinc.		
59	3º	7º	4º	1º	3º	1º	0,902	30,93
56	47º	82º	34º	45º	43º	56º	0,965	24

[+] Existen 286 posibles particiones.

Tabla 5: Ranking y parámetros de selección

Se nota además que si se utilizara Par_A como parámetro de selección, la partición 56 sería elegida, siendo que ésta es en realidad inferior a la partición 59 seleccionada por el parámetro $\rho(H)$.

La Tabla 6 indica las correlaciones entre los valores medidos y los parámetros de selección comparados, tanto en el caso síncrono como en el asíncrono. Se verifica en ambos casos que el radio espectral $\rho(H)$ presenta una mayor correlación que el parámetro Par_A con respecto a los tiempos de procesamiento y número de iteraciones.

	Iteraciones		Tiempo Real		Tiempo de CPU	
	sinc.	asinc.	sinc.	asinc.	sinc.	asinc.
$\rho(H)$	0,414	0,87	0,378	0,343	0,41	0,392
Par_A	0,0377	-0,05	0,0104	0,089	0,0137	0,0137

Tabla 6: Correlaciones

5. CONCLUSIONES

Dado el número de procesadores, los métodos de descomposición tradicionalmente utilizados no poseen dominio sobre el tamaño relativo de los subproblemas, o consideran sistemas muy particulares. Esta carencia dificulta el óptimo aprovechamiento de los sistemas distribuidos, generalmente heterogéneos. El presente trabajo presenta un método que permite descomponer cualquier sistema de ecuaciones con el desbalanceamiento deseado. Para esto, se introduce además un parámetro de selección de particiones que mostra mejor comportamiento experimental que los presentados en trabajos anteriores.

Los resultados experimentales permiten concluir que:

- El método propuesto genera mejores particiones que una selección aleatoria, inclusive en varios casos la óptima.
- Las particiones generadas por el método propuesto son mejores ó iguales que las generadas por la *Descomposición-ε*, con la ventaja adicional de ejercer control sobre el número y tamaño de los subproblemas.
- $\rho(H)$ es un mejor parámetro de selección que el parámetro Par_A.

En resumen, la presente propuesta nos brinda una solución superior a las existentes para la descomposición de problemas con el objeto de resolverlos en un sistema distribuido heterogéneo.

BIBLIOGRAFÍA

[1] Barán B., Kaszkurewicz E. y Bhaya A., "Parallel Asynchronous Team Algorithms: Convergence and Performance Analysis". A ser publicado en la *IEEE Trans. on Parallel & Distributed Systems*. 1996.

[2] Vale M.H., *Descomposição de Redes Eléctricas para Procesamiento Paralelo*. Tesis Doctoral COPPE/UFRJ. Río de Janeiro, Brasil, 1995.

[3] Carré B.A., "Solution of Load-Flow by Partitioning Systems into Trees", *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-88, pp. 1931-1938, noviembre 1968.

[4] Irving M.R. y Sterling M.J.H., "Optical Network Tearing Using Simulated Annealing", *IEEE Proceedings*, vol. 137, no. 1, pp. 69-72, enero 1990.

[5] Gupta M. y Banerjee P., "Demostration of Automatic Data Partitioning Techniques for Parallelizing Compilers on Multicomputers", *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no.2, pp. 179-193, marzo 1992.

[6] Vale M.H., Falcão D.M. y Kaszkurewicz E., "Electrical Power Network Decomposition for Parallel Computations". *IEEE International Symposium on Circuits and Systems*-ISCAS 92. San Diego, California, 1992.

[7] Sezer M. y Šiljak D.D., "Nested epsilon decompositions of complex systems", IFAC" 9th *World Congress*, Budapest, Hungría, julio 1984.

[8] Ikeda M. y Šiljak D.D., "Overlapping decomposition, expansions and contractions of dynamic systems" *Large Scale System 1*, North-Holland Publishing Co., pp.29-38, 1980

[9] Barán B., *Estudo de Algoritmos Combinados Paralelos Assíncronos*. Tesis Doctoral COPPE/UFRJ. Río de Janeiro, Brasil. Octubre 1993.

[10] Barán B., Benítez D., Ramos R., "Partición de Sistemas de Ecuaciones para su Resolución en un Ambiente Computacional Distribuído", *Reporte Técnico del CNC-UNA*, TR-001/96, enero 1996.

[11] Monticelli A., *Fluxo de Carga em Redes de Energia Eléctrica*, Editora Edgar Blücher Ltda, 1983.

Semántica de Sistemas Reactivos. Un preorden entre procesos basado en la ST-bisimulación.

Juan V. Echagüe[1]
echague@fing.edu.uy

Gabriel Fialco[1,2]
gfialco@fing.edu.uy

[1] InCo
Fac. de Ingeniería
U. de la República
Montevideo, Uruguay

[2] ESLAI
Universidad de Luján
Buenos Aires, Argentina

Resumen

En este trabajo definimos un preorden $\sqsubseteq_{ST}$ entre programas inspirado en la ST-bisimulación, donde $G_1 \sqsubseteq_{ST} G_2$ puede interpretarse naturalmente como "G_2 es una implementación paralela de G_1" sobre un modelo semántico "truly concurrent" de sistemas reactivos los Sistemas de Transiciones Asíncronos.

Este preorden es decidible para los sistemas finitos, esta contenido en las equivalencias semánticas que no consideran la información de paralelismo y es compatible con la ST-bisimulación.

Se muestra también que el preorden definido es compatible con una amplia familia de combinadores de procesos, aquellos que pueden definirse en el formato SOS-básico.

Finalmente comparamos $\sqsubseteq_{ST}$ con otras propuestas de la literatura.

Introducción

En el desarrollo de programas paralelos muchas veces poseemos una descripción adecuada del comportamiento deseado del programa, sin que esta incluya información sobre el paralelismo. A la vez, disponemos de hardware paralelo donde implementar dicho programa. Hay en general buenas razones para usar este hardware: en algunos casos, como en el de los protocolos de comunicación, la distribución forma parte del problema, en otros, deseamos simplemente acelerar nuestros cálculos. Surge entonces el propósito de paralelizar un programa (o una especificación). Esto es, a partir de un programa queremos construir otro con "el mismo comportamiento" pero "más paralelo". Este problema ha sido encarado desde múltiples perspectivas, desde la creación de amplias

áreas de estudio, como la de programación paralela, hasta la propuesta de refinadas técnicas de extracción automática de paralelismo.

En este trabajo estudiamos la siguiente pregunta:

¿ Cómo definir cuándo un programa es una implementación paralela de otro?

Nuestra perspectiva es la de la semántica formal (en [Lee90] puede encontrarse una visión de conjunto sobre este tema) buscando, para un cierto modelo semántico de programas que permite representar el paralelismo ("truly concurrent"), una relación $\sqsubseteq$ entre programas tal que $G_1 \sqsubseteq G_2$ pueda interpretarse naturalmente como: G_2 *es una implementación paralela de* G_1. Nos abstraemos así de todo lenguaje de programación, o de especificación o de cualquier arquitectura en particular.

Nuestro objetivo es sentar una base teórica para los Métodos Formales y la Ingeniería de Software necesarias en la construcción de programas paralelos.

Una respuesta a nuestra pregunta desde la semántica formal fue dada por Aceto en [Ace89]. Allí define un preorden $\leq$, "menos paralelo que", inspirado en la bisimulación [Par81] sobre los términos de una versión restringida de CCS [Mil80, Mil89]. Esencialmente, $p \leq q$ cuando todo pomset [Pra86] que p puede ejecutar, puede ser imitado por q con el mismo pomset o, eventualmente, con uno "más paralelo". Recíprocamente, toda ejecución de q puede ser imitada por p, eventualmente, con un pomset "menos paralelo". En todos los casos, la imitación lleva a estados relacionados por $\leq$. Luego de presentar su preorden, Aceto se interesa en las nociones de comporatamiento que este preserva, y que nos dicen en qué sentido un programa es implementación del otro: muestra entonces que los términos relacionados por el preorden son bisimilares. Finalmente provee una axiomatización completa de $\leq$ sobre los términos sin recursión.

Una aproximación más general es presentada por Yankelevich en [Yan93]. Allí propone una técnica para obtener preórdenes entre procesos a partir de preórdenes entre las observaciones de dichos procesos, utilizando distintas formas de la bisimulación. Trabajando con una variante de CCS, el autor muestra varias aplicaciones de esta técnica, entre ellas una para un preorden "más paralelo que", estrictamente más fino que el propuesto por Aceto. Se presentan también axiomatizaciones completas para estos preórdenes, parametrizadas por las axiomatizaciones sobre las observaciones.

Una propuesta similar a la presentada en este trabajo, pero reduciendo el problema al caso determinístico, es abordada por el autor en [Fia95].

En este trabajo abordamos al problema sobre un dominio semántico particular, los *Sistemas de Transiciones Asíncronos* (STA) [Shi85, Bed87]. Los STA son una generalización simple de los Sistemas de Transiciones [Kel76] donde las transiciones están etiquetadas por eventos, sobre los que se define una relación de independencia. Intuitivamente, dos eventos independientes pueden ejecutarse "en paralelo".

La principal contribución de este trabajo es la propuesta de un preorden $\sqsubseteq_{ST}$ sobre los STA, tal que $G_1 \sqsubseteq_{ST} G_2$ puede interpretarse como "G_2 es una implementación

paralela de G_1". Este preorden, inspirado en la ST-bisimulación [GV87], una relación de equivalencia que, al igual que la bismimulación, nos permite formalizar una fina noción de "igual comportamiento", teniendo en cuenta la capacidad de los procesos de ejecutar acciones simultáneamente. Esta definición de ser "más paralelo que" es decidible cuando el STA es finito. Mostramos que $\sqsubseteq_{ST}$ es un preorden compatible con la ST-bisimulación. Establecemos luego que el preorden está contenido en las equivalencias semánticas de la literatura que no consideran el paralelismo, debido a que $\sqsubseteq_{ST}$ sólo compara sistemas bisimilares [Par81, Mil83], esto justifica que lo usemos como noción de "implementación de".

Comparamos este preorden con otros que surgen en la literatura. Demostramos que en el caso determinista es equivalente al presentado en [Fia95], que en el caso no-determinista es estríctamente más grueso que el propuesto en [Yan93], e incomparable con el introducido en [Ace89].

Finalmente, $\sqsubseteq_{ST}$ es una precongruencia para una amplia familia de operadores sobre procesos, todos aqellos que puedan especificarse mediante una semántica operacional estructurada básica.

La organización de este trabajo es la siguiente. En la sección 1 presentamos los STA y la noción de ST-estado (que también llamamos diamante). En la sección 2 presentamos la definición de bisimulación clásica, y la de ST-bisimulación. La sección 3 está dedicada a definir el preorden $\sqsubseteq_{ST}$, y a estudiar sus propiedades. La sección 4 presenta un teorema general de precongruencia para $\sqsubseteq_{ST}$. En la sección 5 comparamos nuestro trabajo con otros que aparecen en la literatura. Concluimos en la sección 6 con una discusión sobre trabajos en curso y futuros.

1 Sistemas de Transiciones Asíncronos

Los *Sistemas de Transiciones Asíncronos (STA)*, introducidos en forma independiente por Shields y Bednarczyk [Shi85, Bed87], son una generalización simple de los Sistemas de Transiciones (ST). Los ST son el modelo mejor conocido de sistemas reactivos [Pnu85]. Son grafos orientados donde los vértices representan los *estados* del sistema (entre ellos generalmente se distingue uno, el *inicial*) y las flechas las *transiciones* entre los estados, *etiquetadas* sobre un alfabeto. Una ejecución de un ST es un camino del grafo que parte de un estado dado, y la observación de esta ejecución es la secuencia de etiquetas que se encuentran sobre las flechas.

Los STA generalizan a los ST: las etiquetas (llamadas *eventos*) poseen una *relación de independencia*. Intuitivamente, dos eventos independientes no comparten recursos y pueden por ello ejecutarse en paralelo. La relación de independencia es una relación binaria, simétrica e irreflexiva I, tal que para toda pareja de eventos $(a, b) \in I$, las transiciones satisfacen:

- *Estabilidad hacia adelante:* Si un estado es origen de dos transiciones etiquetadas

a y b, entonces lo es de dos secuencias etiquetadas $\langle a, b\rangle$ y $\langle b, a\rangle$ que convergen a un mismo estado.

- *Conmutatividad:* Si un estado es origen de una secuencia de transiciones etiquetada $\langle a, b\rangle$, también lo es de una secuencia $\langle b, a\rangle$, y ambas convergen a un mismo estado.

Resumiendo: cuando dos transiciones etiquetadas por eventos independientes parten de un mismo estado o están en secuencia, ellas forman un *diamante*. Esto da nacimiento a la noción de *ST-estado* [GV87], una noción matemática elegante que generaliza las de estado, transición y la de un conjunto de transiciones etiquetadas por eventos independientes. Las transiciones también satisfacen:

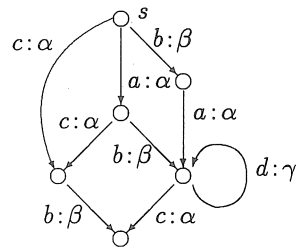
- *Determinismo:* De cada estado parte a lo sumo una transición etiquetada con un evento dado.

Definición 1 *Un* Sistema de Transiciones Asíncrono *(STA) etiquetado sobre un alfabeto de acciones* Act $= \{\alpha, \beta, \gamma, \ldots\}$ *es una estructura* $G = \langle Q, E, I, \rightarrow, l\rangle$, *donde*

- $Q = \{i, s, u, \ldots\}$ *es un* conjunto de estados, *a veces distinguimos a un estado* i *al que llamamos* estado inicial.
- $E = \{a, b, e \ldots\}$ *es un* conjunto de eventos.
- $I \subseteq E \times E$ *es la* relación de independencia, *irreflexiva y simétrica.*
- $\rightarrow \subseteq Q \times E \times Q$, *es la* relación de transición *(escribimos* $s \xrightarrow{e} s_1$ *cuando* $(s, e, s_1) \in \rightarrow$*) que cumple:*
  - *Determinismo. Si* $s \xrightarrow{e} s_1$ *y* $s \xrightarrow{e} s_2$ *entonces* $s_1 = s_2$.
  - *Estabilidad hacia adelante. Si* aIb, $s \xrightarrow{a} s_1$ *y* $s \xrightarrow{b} s_2$ *entonces* $\exists u. s_1 \xrightarrow{b} u \wedge s_2 \xrightarrow{a} u$.
  - *Conmutatividad. Si* aIb *y* $s \xrightarrow{a} s_1 \xrightarrow{b} u$ *entonces* $\exists s_2.\ s \xrightarrow{b} s_2 \xrightarrow{a} u$.
- $l : E \mapsto$ Act *es la función de etiquetado.*

Notamos $\mathcal{A} = \{G, G_1, G_2 \ldots\}$ *la clase de los STA.*

Ejemplo 2 *Mostramos aquí la representación gráfica de un STA sobre el alfabeto* $\{\alpha, \beta, \gamma\}$ *con 6 estados* $E = \{a, b, c, d\}$ *e* $I = \{(a, b), (b, a)\}$.



G

Los ST-estados (o diamantes) son una representación de la ejecución en paralelo de varios eventos independientes a partir de un estado.

Definición 3 (ST-estado) *Sea* $G = \langle Q, i, E, I, \rightarrow \rangle$. *Un* ST-estado *(o* diamante*) de* G *es una pareja* $\langle s, \{e_1, \ldots, e_n\} \rangle \in Q \times \mathcal{P}(E)$ *tal que para todos los* $e_j, e_k \in \{e_1, \ldots e_n\}$, $s \xrightarrow{e_j}$ *y si* $j \neq k$ *entonces* $e_j I e_k$. *Anotamos* $s + \sum_J e_j$ *para referirnos al diamante* $\langle s, \{e_1, \ldots, e_n\} \rangle$, *donde* $J = \{1, \ldots, n\}$ *es el conjunto de índices.*

El destino *de un ST-estado* $\langle s, \{e_1, \ldots e_n\} \rangle$ *(al cual notamos* $\lceil s + \sum_J e_j \rceil$ *siendo* $J = \{1, \ldots, n\}$ *el conjunto de índices) es el estado* u *tal que* $s \xrightarrow{e_1} \ldots \xrightarrow{e_n} u$. *Anotamos* $\mathrm{Diam}(G)$ *el conjunto de ST-estados de* G.

Los ST-estados son una representación abstracta de los estados (por ejemplo $i = \langle i, \emptyset \rangle$), las transiciones (por ejemplo $i \xrightarrow{b} s = \langle i, \{b\} \rangle$) y los diamantes engendrados por eventos independientes (por ejemplo $\langle i, \{a, b\} \rangle$). Nótese que cuando un STA es finito, también lo es el conjunto de sus ST-estados.

2 Bisimulación y ST-bisimulación

Lo que sigue es una presentación algo inusual de la bisimulación clásica de Milner y Park. La presentación apunta a señalar la estrecha relación entre esta definición y las relacionadas a la ST-bisimuación que aparecen más adelante, y sólo difiere de la usual en establecer relaciones entre las transiciones (diamantes de dimensión 1) además de hacerlo, como es usual, sobre los estados (diamantes de dimensión 0).

Esta definición nos dice que dos STA son bisimilares cuando pueden imitarse estados y transiciones.

Definición 4 (Bisimulación) *Sea* $G = \langle Q, E, I, \rightarrow, l \rangle$ *un STA. Una relación* $\mathcal{R} \subseteq Diam(G) \times Diam(G) \times \mathcal{P}(E \times E)$ *es una bisimulación en* G *sii para todo* $(s + \sum_J a_j, r + \sum_K b_k, h) \in \mathcal{R}$ *se cumple que*

1. *(buena formación de* h*)* $h : \{a_j : j \in J\} \rightarrow \{b_k : k \in K\}$ *es una biyección que preserva el etiquetado, i.e.* $l(h(a)) = l(a)$.

2. *(simulación a la derecha hasta una dimensión) si* $J = K = \emptyset$ *y* $s + a \in \mathrm{Diam}(G)$ *entonces* $\exists b \in E$ *tal que* $(r + b \in \mathrm{Diam}(G)$ *y* $(s + a, r + b, \{(a, b)\}) \in \mathcal{R}$.

3. *(simulación a la izquierda hasta una dimensión) si* $J = K = \emptyset$ *y* $r + b \in \mathrm{Diam}(G)$ *entonces* $\exists a \in E$ *tal que* $(s + a \in \mathrm{Diam}(G)$ *y* $(s + a, r + b, \{(a, b)\}) \in \mathcal{R}$.

4. *(culminación)* $\forall i \in J$, $(\lceil s + a_i \rceil + \sum_{J \setminus \{i\}} a_j, \lceil r + h(a_i) \rceil + \sum_{K, b_k \neq h(a_i)} b_k, h\,|_{\{a_j : j \in J, j \neq i\}}) \in \mathcal{R}$

Para todo $s, r \in Q$ *escribimos que* $\mathcal{R} : s \underline{\leftrightarrow} r$ *sii* $\mathcal{R}$ *es una bisimulación de* G *y* $(s, r, \emptyset) \in \mathcal{R}$.

Anotaremos $s \underline{\leftrightarrow} r$ *sii existe* $\mathcal{R}$ *tal que* $\mathcal{R} : s \underline{\leftrightarrow} r$.

Lema 5 *Para todo* $s, r \in Q$, s *es bisimilar con* r *según la definición clásica de bisimulación [Mil83, Par81] sii existe una bisimulación* $\mathcal{R}$ *(según la Definición 4 tal que* $\mathcal{R} : s \underline{\leftrightarrow} r$.

Puede verse que en la definición anterior sólo son trascendentes los diamantes de dimensión 0 y 1 (estados y transiciones), que son los que "deben ser imitados". Podemos hacer la definición más estricta solicitando que la imitación del crecimiento siga hasta cualquier dimensión. Esto nos da exactamente la ST-bisimulación.

Definición 6 (ST-bisimulación [GV87]) *Sea* $G = \langle Q, E, I, \rightarrow, l \rangle$ *un STA. Una relación* $\mathcal{R} \subseteq \mathrm{Diam}(G) \times \mathrm{Diam}(G) \times \mathcal{P}(E \times E)$ *es una ST-bisimulación en* G *sii para todo* $(s + \sum_J a_j, r + \sum_K b_k, h) \in \mathcal{R}$ *se cumple que*

1. *(buena formación de* h*)* $h : \{a_j : j \in J\} \rightarrow \{b_k : k \in K\}$ *es una biyección que preserva el etiquetado, i.e.* $l(h(a)) = l(a)$.

2. *(simulación a la derecha hasta cualquier dimensión) si* $s + \sum_J a_j + a \in \mathrm{Diam}(G)$ *entonces* $\exists b \in E$ *tal que* $r + \sum_K b_k + b \in \mathrm{Diam}(G)$ *y* $(s + \sum_J a_j + a, r + \sum_K b_k + b, h \cup \{(a, b)\}) \in \mathcal{R}$.

3. *(simulación a la izquierda hasta cualquier dimensión) si* $r + \sum_K b_k + b \in \mathrm{Diam}(G)$ *entonces* $\exists a \in E$ *tal que* $s + \sum_J a_j + a \in \mathrm{Diam}(G)$ *y* $(s + \sum_J a_j + a, r + \sum_K b_k + b, h \cup \{(a, b)\}) \in \mathcal{R}$.

4. *(culminación)* $\forall i \in J$, $(\lceil s + a_i \rceil + \sum_{J \setminus \{i\}} a_j, \lceil r + h(a_i) \rceil + \sum_{K, b_k \neq h(a_i)} b_k, h\,|_{\{a_j : j \in J, j \neq i\}}) \in \mathcal{R}$

Para todo $s, r \in Q$ *escribimos que* $\mathcal{R} : s \equiv_{ST} r$ *sii* $\mathcal{R}$ *es una ST-bisimulación de* G *y* $(s, r, \emptyset) \in \mathcal{R}$.

Diremos que s *y* r *son ST-bisimilares, lo que anotaremos* $s \equiv_{ST} r$ *sii existe* $\mathcal{R}$ *tal que* $\mathcal{R} : s \equiv_{ST} r$.

3 ST-Prebisimulación

En la sección anterior hemos presentado dos equivalencias sobre los STA. Ambas pueden verse como un elaborado juego de simulación entre dos procesos, donde cada uno debe poder imitar al otro, y el rol de imitador e imitado pueden intercambiarse en cada momento. La diferencia entre esas dos equivalencias reside en cuáles son las cosas que deben imitarse: la bisimulación es, en ese sentido, menos exigente que la ST-bisimulación, ya que pide solamente imitar estados y transiciones, esto es, ST-estados de dimensión 0 y 1. La ST-bisimulación exige que esta imitación se mantenga a cualquier dimensión.

En esta sección presentaremos la primera contribución de nuestro trabajo: definiremos la ST-prebisimulación sobre los STA. Intuitivamente la ST-prebisimulación es también un juego de simulación donde los roles de imitador e imitado pueden intercambiarse en cada instante. Pero los jugadores tendrán diferente exigencia respecto a aquello que debe imitarse. El proceso "más paralelo" debe imitar todo aquello que el "menos paralelo" pueda hacer, pero la exigencia para éste es más débil: basta que pueda imitar, como en la bisimulación clásica, los estados y las transiciones.

Puede pensarse que es un juego de imitación, con intercambio de roles, entre niños de diferente edad: aquel que es mayor podrá imitar todo lo que el más pequeño haga, y éste aquellos gestos más simples de su compañero mayor.

Luego de presentar la definición de la ST-bisimulación presentaremos algunas de sus propiedades esenciales, entre otras, que forma un preorden sobre los STA, y que tiene buenas relaciones con la bisimulación y la ST-bisimulación.

Definición 7 (ST-prebisimulación) *Sea* $G = \langle Q, E, I, \rightarrow, l\rangle$ *un STA. Una relación* $\mathcal{R} \subseteq Diam(G) \times Diam(G) \times \mathcal{P}(E \times E)$ *es una ST-prebisimulación de* G *sii para todo* $(s + \sum_J a_j, r + \sum_K b_k, h) \in \mathcal{R}$ *se cumple que*

1. *(buena formación de h)* $h : \{a_j : j \in J\} \rightarrow \{b_k : k \in K\}$ *es una biyección que preserva el etiquetado, i.e.* $l(h(a)) = l(a)$.

2. *(simulación a la derecha hasta cualquier dimensión) si* $s + \sum_J a_j + a \in \mathrm{Diam}(G)$ *entonces* $\exists b \in E$ *tal que* $r + \sum_K b_k + b \in \mathrm{Diam}(G)$ *y* $(s + \sum_J a_j + a, r + \sum_K b_k + b, h \cup \{(a, b)\}) \in \mathcal{R}$.

3. *(simulación a la izquierda hasta una dimensión) si* $J = K = \emptyset$ *y* $r + b \in \mathrm{Diam}(G)$ *entonces* $\exists a \in E$ *tal que* $(s + a \in \mathrm{Diam}(G)$ *y* $(s + a, r + b, \{(a, b)\}) \in \mathcal{R}$.

4. *(culminación)* $\forall i \in J, (\lceil s + a_i\rceil + \sum_{J\setminus\{i\}} a_j, \lceil r + h(a_i)\rceil + \sum_{K, b_k \neq h(a_i)} b_k, h\mid_{\{a_j : j\in J, j\neq i\}}) \in \mathcal{R}$

Para todo $s, r \in Q$ *escribimos que* $\mathcal{R} : s \sqsubseteq_{ST} r$ *sii* $\mathcal{R}$ *es una ST-prebisimulación de* G *y* $(s, r, \emptyset) \in \mathcal{R}$.

Diremos que s *y* r *son ST-prebisimilares, lo que anotaremos* $s \sqsubseteq_{ST} r$ *sii existe* $\mathcal{R}$ *tal que* $\mathcal{R} : s \sqsubseteq_{ST} r$.

Lema 8 *Sean G un STA y $\mathcal{R}_1$ y $\mathcal{R}_2$ dos ST-prebisimulaciones de G, entonces $\mathcal{R}_1 \cup \mathcal{R}_2$ es una ST-prebisimulación de G.*

Corolario 9 *Sea G un STA, entonces existe $\mathcal{R}$ la mayor ST-prebisimulación de G.*

La composición de dos ST-prebisimulaciones se puede definir de la siguiente manera:

Definición 10 *Sean $\mathcal{R}_1$ y $\mathcal{R}_2$ dos ST-prebisimulaciones de un STA G dado, entonces*

$$\mathcal{R}_2 \circ \mathcal{R}_1 = (s + \sum_J a_j, r + \sum_K b_k, h_2 \circ h_1) \mid \text{exiten}$$
$$(s + \sum_J a_j, t + \sum_I c_i, h_1) \in \mathcal{R}_1 \text{ y } (t + \sum_I c_i, r + \sum_K b_k, h_2) \in \mathcal{R}_2$$

Se cumple entonces el siguiente lema:

Lema 11 *Sean G un STA dado y $\mathcal{R}_1$ y $\mathcal{R}_2$ dos ST-prebisimulaciones en G, entonces $\mathcal{R}_2 \circ \mathcal{R}_1$ también es una ST-prebisimulación en G.*

La composición de una ST-bisimulación con una ST-prebisimulación , y la de una ST-prebisimulación con una ST-bisimulación se pueden definir en forma totalmente análoga a la Definición 10.

Se cumplen entonces los siguientes dos lemas:

Lema 12 *Sean G un STA dado, $\mathcal{R}_1$ una ST-prebisimulación en G y $\mathcal{R}_2$ una ST-bisimulación en G, entonces $\mathcal{R}_2 \circ \mathcal{R}_1$ es una ST-prebisimulación en G.*

Lema 13 *Sean G un STA dado, $\mathcal{R}_1$ una ST-bisimulación en G y $\mathcal{R}_2$ una ST-prebisimulac en G, entonces $\mathcal{R}_2 \circ \mathcal{R}_1$ es una ST-prebisimulación en G.*

Por último, presentaremos algunas propiedades de $\sqsubseteq_{ST}$.

Lema 14 *$\sqsubseteq_{ST} \subseteq Q \times Q$ es un preorden.*

Lema 15 *Sean G un STA y $s, r, s', r' \in Q$ estados de G tales que $s' \equiv_{ST} s \sqsubseteq_{ST} r \equiv_{ST} r'$, entonces $s' \sqsubseteq_{ST} r'$.*

Lema 16 *Sean G un STA y $s, r \in Q$ dos estados de G tales que $s \sqsubseteq_{ST} r$ entonces $s \leftrightarrow r$.*

4 Teorema general de precongruencia

En esta sección presentaremos un teorema que nos habla de como interactúan la relación de ST-prebisimulación recién definida y los combinadores de procesos usuales. El resultado buscado en general es que la relación definida nos permite razonar modularmente sobre los procesos construidos por los combinadores. Esto quiere decir que deseamos mostrar que en general, dado un operador *op*, el hecho que $p \sqsubseteq_{ST} q$ nos basta para asegurar que $op(p) \sqsubseteq_{ST} op(q)$.

El teorema no se plantea para un lenguaje en particular. El resultado que presentamos aquí se cumple para una gran familia de operadores, todos aquellos que se encuentran en lenguajes que pueden describirse con la ayuda de la técnica SOS utilizando un cierto formato particular llamado "de base" en la literatura. En realidad este formato es suficientemente amplio para definir las álgebras de procesos más conocidas, como CCS [Mil80, Mil89]. Y es el formato más libre para el que se conoce una forma canónica de construir un STA a partir de una descripción SOS [BD92, EHP95].

Teorema 17 (Precongruencia) *La ST-prebisimulación es una precongruencia para todos los operadores definidos en una especificación SOS básica irreflexiva.*

En otras palabras, sea G_P el STA asociado a una especificación irreflexiva P y $\mathcal{R}_P$ la mayor ST-prebisimulación en G_P. Entonces, para todo $t_i, u_i \in T_\Sigma$ con $i = 1, \ldots, n$ y $f \in \Sigma^n$ se cumple que

$$\forall i = 1, \ldots, n. \mathcal{R}_P : t_i \sqsubseteq_{ST} u_i \Rightarrow \mathcal{R}_P : f(t_1, \ldots, t_n) \sqsubseteq_{ST} f(u_1, \ldots, u_n)$$

5 Comparación con otras propuestas

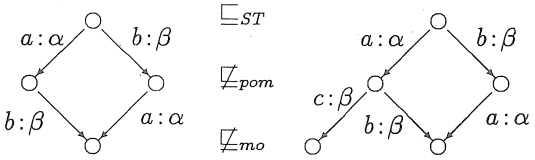
En esta sección comparamos el preorden $\sqsubseteq_{ST}$ definido en este trabajo con otros preórdenes que, buscando formalizar la noción de "más paralelo que", aparecen en la literatura.

A nuestro conocimiento existen tres propuestas relacionadas con la que aquí se presenta, el preorden presentado por Aceto en [Ace89] (inspirado en la *pomset bisimulation* [BC87, BC88], que aquí llamaremos $\sqsubseteq_{pom}$), el preorden $\sqsubseteq_{mo}$ introducido por Yankelevich en [Yan93] (inspirado en la *mixed order equivalence* [DDM89]) y el preorden $\sqsubseteq$ propuesto por el autor en [Fia95] (para STA deterministas).

El resultado es que $\sqsubseteq_{ST}$ es incomparable con $\sqsubseteq_{pom}$, mientras que es más gruesa que $\sqsubseteq_{mo}$ (inclusive considerando esta para el caso en que no aparece la acción sigilosa τ, que es el que aquí consideramos). En el caso de los STA deterministas, la propuesta presentada aquí coincide con $\sqsubseteq$.

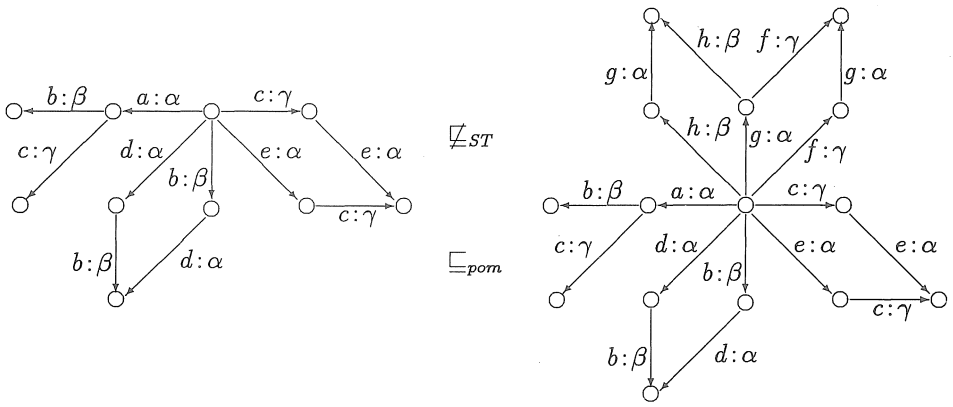
Los siguientes ejemplos nos permiten demostrar la relación que guarda nuestro trabajo con el trabajo de Aceto y parte de la relación que este guarda con el de Yankelevich. Del ejemplo

Ejemplo 18 *Mostramos aquí que $\sqsubseteq_{ST} \not\subseteq \sqsubseteq_{pom}$ y que $\sqsubseteq_{ST} \not\subseteq \sqsubseteq_{mo}$. En este ejemplo aIb.*



podemos concluir que $\sqsubseteq_{ST}$ no esta incluida ni en $\sqsubseteq_{pom}$ ni en $\sqsubseteq_{mo}$. Del ejemplo

Ejemplo 19 *Mostramos aquí que* $\sqsubseteq_{pom} \not\subseteq \sqsubseteq_{ST}$. *En este ejemplo* $I = \{(d,b),(e,c),(g,f),(g,h)$



podemos concluir que $\sqsubseteq_{pom}$ no esta incluido en $\sqsubseteq_{ST}$.

Puede demostrarse que $\sqsubseteq_{mo}$ es estríctamente más fino que $\sqsubseteq_{ST}$, y que en el caso determinista $\sqsubseteq_{ST}$ coincide con $\sqsubseteq$. Desafortunadamente para presentar las pruebas de estas propiedades es necesario agregar una buena cantidad de nuevas definiciones, por lo que hemos preferido simplemente enunciarlas. No es un trabajo complejo, pero si extenso.

6 Conclusiones

En este trabajo formalizamos una noción de "un programa es una implementación paralela de otro" en el marco de los Sistemas de Transiciones Asíncronos de manera independiente de cualquier lenguaje de programación, especificación o arquitectura.

Hemos definido para ello el preorden $\sqsubseteq_{ST}$ en los STA. Cuando $G_1 \sqsubseteq_{ST} G_2$ ambos hacen las mismas cosas (son equivalentes en todas las equivalencias semánticas de la literatura que no toman en cuenta el paralelismo) y a la vez G_2 lo hace de manera más paralela (los eventos que pueden ocurrir en paralelo en G_1 pueden hacerlo en G_2). Mostramos que si dos STA están relacionados por $\sqsubseteq_{ST}$ entonces son bisimilares, lo que nos asegura que tienen el mismo comportamiento. Probamos además que $\sqsubseteq_{ST}$ es compatible con la ST-bisimulación, en el sentido que preserva las clases de equivalencia. Por otro lado, al estar definido como una relación entre ST-estados, si el STA es finito $\sqsubseteq_{ST}$ es trivialmente decidible.

Nuestro preorden es estrictamente más grueso que el propuesto por Yankelevich en [Yan93], e incomparable con el preorden presentado por Aceto en [Ace89]. Presentamos ejemplos que evidencian estos resultados. Por otro lado, si reducimos nuestra definición al caso determinista, $\sqsubseteq_{ST}$ coincide con el presentado por el autor en [Fia95].

Por último, demostramos que, para una amplia familia de operadores de procesos, más precisamente, todos aquellos que puedan definirse a través de una semántica operacional estructurada básica [BIM88], $\sqsubseteq_{ST}$ resulta ser una precongruencia.

Existen varias líneas para continuar esta investigación. Una es ampliar nuestra definición a los STA con la acción invisible τ. Otra, sobre la cual estamos formando un grupo de trabajo, es seguir la línea de Aceto y buscar una axiomatización para el preorden.

Agradecimientos

Agradecemos a Pedro Ruben D'Argenio, Juan José Cabezas, Guillermo Calderón, Victoria Galatro, Martina Marré, Alfredo Olivero, Cecilia Pertino, Luis Sierra, Jorge Vidart, Daniel Yankelevich y Sergio Yovine que nos han hecho valiosos comentarios.

Este trabajo ha sido parcialmente financiado por el *Programa de Desarrollo de Ciencias Básicas*, (PEDECIBA), Uruguay.

Referencias

[Ace89] L. Aceto. On relating concurrency and nondeterminism. Report 6/89, Univ. Sussex, Brighton, GB, October 1989.

[BC87] G. Boudol and I. Castellani. On the semantics of concurrency: Partial orders and transition systems. In *Proc. CAAP'87, Pisa, LNCS 249*, pages 123–137. Springer-Verlag, March 1987.

[BC88] G. Boudol and I. Castellani. Concurrency and atomicity. *Theoretical Computer Science*, 59(1):25–84, 1988.

[BD92] E. Badouel and Ph. Darondeau. Structural operational specifications and trace automata. In *Proc. CONCUR'92, Stony Brook, NY, LNCS 630*, pages 302–316. Springer-Verlag, August 1992.

[Bed87] M. A. Bednarczyk. *Categories of Asynchronous Systems.* PhD thesis, Univ. Sussex, October 1987. Available as CS R 1/88.

[BIM88] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced: preliminary report. In *Proc. 15th ACM Symp. Principles of Programming Languages, San Diego, CA*, pages 229–239, January 1988.

[DDM89] P. Degano, R. De Nicola, and U. Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 438–466. Springer-Verlag, 1989.

[EHP95] J. Echagüe, Z. Habbas, and S. Pinchinat. Structural Operational Semantics Specifications for True Concurrency., 1995.

[Fia95] G. Fialco. Cuándo un programa es más paralelo que otro? Pasantía, Escuela Superior Latinoamericana de Informática, 1995.

[GV87] R. J. van Glabbeek and F. Vaandrager. Petri net models for algebraic theories of concurrency. In *Proc. PARLE'87, vol. II: Parallel Languages, Eindhoven, LNCS 259*, pages 224–242. Springer-Verlag, June 1987.

[Kel76] R. M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, July 1976.

[Lee90] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science, vol. B.* Elsevier Science Publishers, 1990.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 23(3):267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall Int., 1989.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proc. 5th GI Conf. on Th. Comp. Sci., LNCS 104*, pages 167–183. Springer-Verlag, March 1981.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Lect. Notes, Aarhus University, Aarhus, DK, 1981.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th ICALP, Nafplion, LNCS 194*, pages 15–32. Springer-Verlag, July 1985.

[Pra86] V. R. Pratt. Modeling concurrency with partial orders. *Int. J. Parallel Programming*, 15(1):33–71, 1986.

[Shi85] M. W. Shields. Concurrent machines. *The Computer Journal*, 28(5):449–465, 1985.

[Yan93] D. N. Yankelevich. *Parametric Views of Process Description Languages.* PhD thesis, Univ. Pisa, 1993.

YALI
Uma Extensão do Modelo Linda para Programação Paralela em Redes Heterogêneas

Andréa Schwertner Charão
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre – RS – Brasil
e-mail: andrea@inf.ufrgs.br

Celso Maciel da Costa
Instituto de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre – RS – Brasil
e-mail: celso@kriti.pucrs.br

Sumário

Com a disponibilidade de redes que ligam estações cada vez mais poderosas a baixos custos, o interesse em torno de ambientes que suportam a programação paralela em arquiteturas deste tipo tem aumentado muito. Este artigo descreve um ambiente destinado à programação paralela em redes heterogêneas, baseado no modelo Linda. Este ambiente, chamado YALI, suporta operações globais, que auxiliam na comunicação e sincronicação entre múltiplos processos. YALI também inclui um espaço de tuplas distribuído, baseado em *hashing*, e implementado com *threads*.

Palavras-chave: Linda, programação paralela, espaço de tuplas.

Abstract

With the availability of networks connecting powerful workstations at a low cost, increasing interest has been devoted to systems that support parallel programming in such architectures. This paper describes an environment for parallel programming in heterogeneous networks, based on the Linda model. This parallel environment, called YALI, provides some global operations, that are useful to express communication and synchronization between multiple processes. YALI also includes a multithreaded, distributed tuple space, based on hashing.

Keywords: Linda, parallel programming, tuple space.

1 Introdução

Existe, atualmente, um crescente interesse em torno de ambientes que suportam a programação paralela em redes de computadores, devido principalmente à disponibilidade de redes ligando estações e poderosos computadores pessoais, que reúnem um grande poder computacional a custos relativamente baixos. Nestes ambientes, o paradigma de programação mais frequentemente empregado é baseado em troca de mensagens, que, apesar de ser eficiente, é de difícil assimilação devido ao seu baixo nível de abstração.

Uma abordagem alternativa à troca de mensagens é permitir a comunicação entre processos através de uma memória logicamente compartilhada. Dentro desta abordagem de mais alto nível, pode-se destacar o modelo Linda[1][GEL85], onde toda interação entre processos ocorre por intermédio de uma memória compartilhada associativa, denominada espaço de tuplas. Entre as vantagens deste modelo pode-se citar a simplicidade de suas primitivas, e a possibilidade de incorporá-las a uma linguagem sequencial conhecida.

Este artigo descreve YALI, um ambiente para programação paralela em redes heterogêneas. Tendo como objetivo facilitar a programação paralela em arquiteturas deste tipo, YALI baseia-se no modelo Linda, mas possui extensões para suporte a operações globais[FOS95], usadas para comunicação e sincronização envolvendo vários processos. O paralelismo, em YALI, é baseado em processos distribuídos de modo estático, e *threads* disparadas dinamicamente. O espaço de tuplas usado em YALI é distribuído entre os processos que compõem uma aplicação paralela, e sua implementação é baseada em *threads*. O artigo está organizado da seguine maneira: a seção 2 discute sucintamente o modelo Linda e alguns aspectos de sua implementação. Na seção 3, a seguir, são apresentados os componentes do ambiente YALI. As seções 4 e 5, por fim, tratam da implementação e da avaliação do ambiente, respectivamente.

2 O Modelo Linda

O modelo Linda[GEL85] consiste em um pequeno conjunto de primitivas que manipulam uma memória global associativa denominada **espaço de tuplas** (*Tuple Space* — TS) . As primitivas Linda, quando incorporadas a uma linguagem sequencial, permitem expressar a comunicação e a sincronização entre processos, tornando-a adequada para programação paralela. Basicamente, existem quatro operações definidas em Linda: **out**, **in**, **rd** e **eval**. Todas elas são processadas atomicamente e manipulam entidades chamadas **tuplas**, que são sequências de campos de tipos específicos, aos quais podem ou não ser associados valores. Os tipos dos campos geralmente são aqueles suportados pela linguagem hospedeira de Linda, e a presença ou não de valores associados a um campo define, respectivamente, se o campo é **real** ou **formal**.

De maneira geral, as primitivas servem para inserir ou retirar tuplas do TS. Uma operação **out** insere assincronamente uma tupla no espaço, enquanto **eval** dispara um novo processo para processar os campos de uma tupla, depositando a tupla resultado no TS.

[1]Linda é marca registrada de Scientific Computing Associates, Inc.

A primitiva **in**, ao contrário, serve para remover uma tupla do espaço. A tupla fornecida como argumento à primitiva **in** é chamada *template*, e pode conter campos reais, que formam uma espécie de chave para busca da tupla, e campos formais, representados por variáveis que devem receber algum valor depois que uma tupla equivalente é encontrada no TS. Caso tal tupla não exista no TS, o processo que executa **in** é bloqueado até que uma tupla seja depositada. De maneira semelhante atua a primitiva **rd**, que também serve para a recuperação de tuplas, porém sem removê-las do TS. Existem também duas variações não-bloqueantes das primitivas **in** e **rd**, chamadas, respectivamente, **inp** e **rdp**[NAR89].

2.1 Aspectos de Implementação do Modelo

Apesar da simplicidade associada ao modelo Linda, sua implementação eficiente depende de alguns fatores. O espaço de tuplas, uma das principais características do modelo, é facilmente implementado em arquiteturas paralelas com memória compartilhada. No entanto, em arquiteturas desprovidas deste recurso, como é o caso das redes de computadores, é necessária a implementação de mecanismos que permitam o compartilhamento lógico do TS. Para isso, muitos sistemas utilizam um processo gerenciador do espaço de tuplas, que processa requisições de operações Linda. Este processo pode estar centralizado em somente um nodo da rede (como em Glenda[SEY93] e p4-Linda[BUT93]), instanciado em alguns nodos (D-Linda[PIN91]) ou mesmo replicado em toda a rede (POSYBL[SCH91]). Embora um gerenciador centralizado seja mais simples de implementar, sistemas Linda que adotam esta solução geralmente não são escaláveis. Espaços de tuplas distribuídos, onde vários gerenciadores estão envolvidos, suportam melhor o desenvolvimento de aplicações com grande número de processos espalhados sobre vários nodos.

Ao implementar um espaço de tuplas distribuído, deve-se adotar alguma política de distribuição das tuplas entre os diversos gerenciadores instanciados em diversos nodos. As políticas mais frequentemente empregadas são as seguintes:

- Distribuição uniforme[GEL85]: segundo este esquema, cada tupla gerada através de **out** é armazenada em um conjunto pré-determinado de nodos, denominado *out_set*. Tuplas requisitadas através de **in**, por sua vez, são procuradas num conjunto de nodos denominado *in_set*. Deve-se garantir que cada *in_set* tenha uma intersecção não vazia com os conjuntos *out_set*. Esta política admite inúmeras possibilidades. Em S/Net[CAR86], por exemplo, todos os nodos fazem parte do *out_set*, e o *in_set* consiste apenas no nodo que executa **in** ou **rd**, o que caracteriza um espaço de tuplas totalmente replicado. O contrário também é possível, onde o *out_set* contém apenas o nodo que executa **out**, e o *in_set* engloba todos os nodos da rede, como em POSYBL[SCH91].

- *Hashing*: neste esquema, o nodo onde uma tupla deve ser armazenada ou procurada é determinado através de uma função de *hashing* aplicada a uma chave extraída da tupla. Esta chave pode ser determinada de diversas maneiras, e tuplas com diferentes chaves são armazenadas em nodos distintos. Um exemplo de implementação baseada em *hashing* é C-Linda[CAR94].

Outra questão associada à implementação de Linda diz respeito à interface oferecida ao usuário. Muitas implementações definem uma biblioteca de primitivas Linda, que se encarregam da interação com os gerenciadores de espaços de tuplas. Esta solução é empregada em várias implementações, como POSYBL[SCH91] e Glenda[SEY93]. Uma alternativa ao uso de uma biblioteca é a construção de um novo compilador para uma linguagem sequencial extendida com as primitivas Linda, como em C-Linda[CAR94]. Esta solução permite desenvolver uma interface mais simples para o usuário, porém sua implementação é mais trabalhosa.

3 O Ambiente YALI

YALI (*Yet Another Linda Implementation*) é um ambiente para programação paralela em redes heterogêneas, baseado no modelo Linda. Além das primitivas propostas neste modelo, que são incorporadas à linguagem C, YALI suporta também algumas operações globais, usadas para comunicação e sincronização envolvendo múltiplos processos, que são necessárias em vários tipos de aplicações, mas geralmente trabalhosas de expressar em Linda.

A fim de garantir flexibilidade ao programador, YALI permite estruturar uma aplicação paralela tanto num modelo SPMD (*Single Program Multiple Data*), onde um mesmo programa executa em cada nodo da rede (embora processando trechos de código diferentes), como também segundo o modelo de processos comunicantes ou MPMD (*Multiple Program Multiple Data*), onde vários processos diferentes formam a aplicação paralela.

O espaço de tuplas utilizado em YALI é distribuído segundo uma política baseada em *hashing*. Ao contrário da maioria das implementações Linda, YALI não emprega um processo especial para gerenciamento do espaço de tuplas, fazendo com que cada processo seja responsável pelo gerenciamento de uma porção do espaço global.

Para dar suporte ao desenvolvimento e execução de aplicações, YALI conta com três componentes distintos: uma biblioteca de funções (YaliLib), um pré-processador (YaliPP), e um programa de inicialização de aplicações paralelas (YaliStart).

3.1 A Biblioteca YaliLib

As funções desta biblioteca constituem a interface com o programador e, em sua maioria, implementam as operações sobre o espaço de tuplas previstas em Linda. Cinco destas operações têm o comportamento idêntico àquele proposto no modelo Linda, e por isso não serão discutidas em detalhe. São elas: Y_OUT, Y_IN, Y_RD, Y_INP, Y_RDP. As primitivas complementares ou que não seguem estritamente o modelo serão apresentadas a seguir.

- Y_EVAL: embora proposta pelo modelo Linda, esta primitiva tem uma semântica um pouco diferente em YALI, pois não está automaticamente associada à inserção de tuplas. Neste ambiente, Y_EVAL serve para disparar uma nova *thread*, que de-

verá executar uma função especificada. Esta função deve ser implementada pelo processo que a invoca através de Y_EVAL.

- Y_GLOBAL: esta primitiva permite associar um nome global a uma função implementada por um determinado processo. Através desta associação, qualquer processo pode disparar a execução da função, bastando, para isso, conhecer seu nome global.
- Y_GLOBEVAL: usada em conjunto com Y_GLOBAL, esta função atua como uma chamada remota de procedimento, fazendo com que a função associada ao nome global especificado seja processada por uma nova *thread*, disparada pelo processo que implementa a função.
- Y_REDUCE: esta primitiva é semelhante a Y_IN, porém permite remover um certo número de tuplas que satisfazem um determinado *template* e, adicionalmente, executar operações simples sobre os diversos valores coletados para os campos formais deste *template*. As operações são pré-definidas em YALI, recebem somente um argumento e produzem um resultado do mesmo tipo do argumento. Exemplos de operações são SUM, que retorna a soma dos valores dos campos, e MIN, que retorna o menor dos valroes encontrados. Y_REDUCE é uma primitiva bloqueante, isto é, o processo (ou *thread*) que a executa é interrompido até que o número especificado de tuplas seja encontrado.
- Y_BARRIER: esta primitiva é usada para sincronizar a execução de um grupo de processos. Cada processo que a utiliza permanece bloqueado até que o número especificado de processos tenha também executado Y_BARRIER.
- Y_ID: retorna a identificação do processo. O identificador é um número inteiro de 0 a (N-1), sendo N o número de processos da aplicação. Cada processo tem um identificador único, que pode ser usado para selecionar o código a ser executado em programas SPMD.
- Y_NPROC: esta primitiva retorna o número de processos que compõem uma aplicação paralela.

Para ilustrar a utilização de algumas primitivas YALI, a figura 1 mostra a implementação de uma aplicação paralela simples, composta por dois tipos de processos. um mestre, que gera tarefas, e um trabalhador, que as executa. Deve-se notar, nos *templates* utilizados com Y_IN e Y_REDUCE, a distinção entre campos reais e formais: estes últimos são precedidos pelo caracter '?'.

3.2 O Pré-Processador e o Programa de Inicialização

O pré-processador YaliPP tem o propósito de converter as chamadas YALI para um formato utilizado internamente pela biblioteca. Além disso, ele mantém uma tabela de símbolos para identificar automaticamente os tipos dos campos das tuplas fornecidas como parâmetro para Y_OUT, Y_IN, Y_RD, Y_INP, Y_RDP e Y_REDUCE, o que facilita a descrição dos campos pelo programador.

Processo Mestre

```
#define N 10

yali_main() {

  int i, result;

  for (i = 0; i < N; i++) {
    y_out("work", i);
    y_globeval("worker");
  }

  y_reduce(N, "partial_result", sum(?result));
  y_barrier("end", N);
}
```

Processo Trabalhador

```
#define N 10

yali_main() {

  y_global("worker", (void *(*)())worker());
  y_barrier("end", N+1);

}
worker() {

  int work, result;

  y_in("work", ?work);
  /* processa tarefa */
  y_out("partial_result", result);
}
```

Figura 1: Exemplo de aplicação YALI.

A inicialização de uma aplicação YALI é feita através de um programa chamado YaliStart. Este programa recebe do usuário a especificação dos processos da aplicação paralela e, opcionalmente, das máquinas onde a aplicação será executada. Esta especificação pode ser feita de três modos. No primeiro, o usuário fornece na linha de comando as informações para execução da aplicação. Este método deve ser utilizado para aplicações simples, com poucos processos diferentes (modelo SPMD, por exemplo). No segundo modo, mais genérico e flexível, utiliza-se um arquivo de configuração. Alternativamente, para usuários pouco experientes, a configuração pode ser especificada interativamente.

O programa de inicialização é responsável por distribuir os processos sobre a rede, e garantir que estes poderão se comunicar uns com os outros. A criação de processos só pode ser feita durante a inicialização, não sendo permitida a criação dinâmica de processos, mas apenas de *threads*, através de Y_EVAL ou Y_GLOBEVAL.

4 Implementação do Ambiente

A seguir serão descritos alguns detalhes sobre a implementação do espaço de tuplas e das primitivas YALI, juntamente com os procedimentos de inicialização e término de aplicações.

4.1 Arquitetura do Espaço de Tuplas

Como mencionado na seção anterior, o espaço de tuplas em YALI é distribuído segundo uma política de *hashing*, e cada processo que compõe uma aplicação é responsável por gerenciar uma porção deste espaço global. Esta arquitetura torna mais eficiente a comunicação entre processos YALI, já que mensagens para manutenção do TS são trocadas diretamente, sem o intermédio de um processo gerenciador de espaço de tuplas.

Esta arquitetura do espaço de tuplas é implementada através de duas *threads* especiais

em cada processo, como pode ser visto na figura 2. Uma das *threads*, chamada *thread* **gerenciadora**, processa operações YALI requisitadas tanto local (por uma *thread* do próprio processo) como remotamente (por *threads* de outros processos no mesmo nodo ou em nodos diferentes). Esta *thread* permanece bloqueada até que uma requisição seja inserida num *buffer* global de requisições (ver figura 2). A segunda *thread*, denominada *thread* de **comunicação**, permanece bloqueada até que uma requisição de operação seja recebida através da rede. Ela é responsável por inserir a requisição no *buffer* global, sinalizando à *thread* gerenciadora que existe nova requisição a processar.

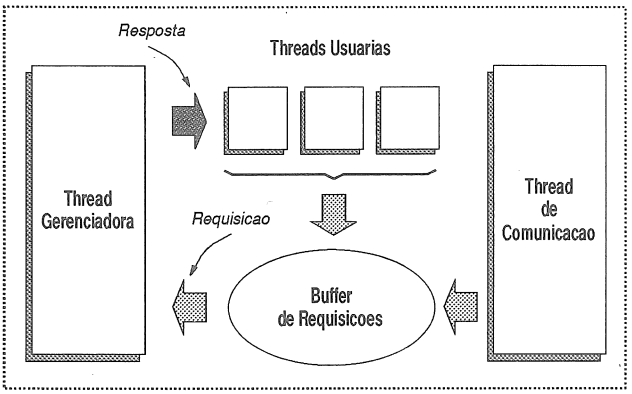


Figura 2: Processo YALI.

YALI permite que várias *threads* utilizem primitivas bloqueantes ao mesmo tempo, porque são usados semáforos diferentes para bloquear cada *thread*. As *threads* usuárias se comunicam com a *thread* gerenciadora depositando requisições no *buffer* global, como pode ser visto na figura 2. Toda requisição contém o endereço de memória onde deve ser colocada a resposta correspondente. Para tratar adequadamente cada requisição, a *thread* gerenciadora mantém as seguintes estruturas de dados:

- o espaço de tuplas propriamente dito, implementado como uma tabela *hash*;
- uma tabela de requisições Y_IN, Y_RD e Y_REDUCE pendentes, onde são armazenados *templates* cujas tuplas equivalentes não foram produzidas;
- uma tabela para gerência de barreiras, onde são armazenadas informações sobre processos bloqueados numa determinada barreira;
- uma tabela para gerência de funções globais, onde é mantida a identificação dos processos que implementam determinadas funções tornadas globais e
- uma tabela com informações sobre os processos que compõem a aplicação (identificação, conexões abertas, etc.).

As *threads* de comunicação trocam mensagens através de protocolos suportados em sistemas Unix. As mensagens constituem requisições ou respostas de operações, geral-

mente contendo tuplas ou *templates*. Processos no mesmo nodo se comunicam através de datagramas ou conexões no domínio Unix, enquanto processos remotos se comunicam através dos protocolos UDP ou TCP. Mensagens com até 2Kbytes são transferidas através de datagramas ou UDP, e mensagens maiores causam o estabelecimento dinâmico de uma conexão (Unix ou TCP) entre os processos comunicantes. Uma conexão, depois de estabelecida, é usada para transmissão de quaisquer mensagens entre os dois processos envolvidos, aproveitando melhor o custo da conexão. Para suportar a comunicação em redes heterogêneas, toda mensagem inclui uma identificação da arquitetura onde foi originada. Quando as arquiteturas são diferentes, o receptor se encarrega da decodificação da mensagem.

4.2 Implementação das Primitivas

Como a política de distribuição de tuplas é baseada em *hashing*, toda primitiva de manipulação de tuplas gera uma mensagem de requisição ao processo que mantém a tupla especificada. A identificação do processo é obtida pela aplicação da função de *hashing* sobre uma chave formada pela descrição de cada campo da tupla. Esta política de distribuição é altamente escalável, pois o número máximo de mensagens trocadas não aumenta com o número de processos. Para a implementação de Y_OUT, por exemplo, é necessário no máximo uma mensagem para enviar a tupla ao processo determinado através de *hashing*. Da mesma maneira, Y_IN, Y_RD, Y_INP e Y_RDP exigem, no máximo, uma mensagem de requisição e outra de resposta, caso a tupla solicitada não seja mantida pelo próprio processo. No caso de Y_IN e Y_RD, suas implementações envolvem ainda o registro de uma pendência para cada requisição que não for satisfeita. Quando a tupla que satisfaz uma pendência é gerada com Y_OUT, a requisição é respondida, e a pendência removida. Já as implementações de Y_INP e Y_RDP não requerem a manutenção de pendências, pois as primitivas não são bloqueantes.

As únicas primitivas que não são implementadas com auxílio de *hashing* são Y_EVAL, Y_ID e Y_NPROC. Y_EVAL apenas causa o disparo de uma nova *thread*, enquanto Y_ID e Y_NPROC retornam valores recebidos do processo YaliStart durante a inicialização da aplicação. A seguir será discutida a implementação das primitivas restantes.

- Y_GLOBAL: para tornar o nome de uma função globalmente acessível, o endereço da função e a identificação do processo que a implementa são enviados para um determinado processo da aplicação, onde serão mantidos numa tabela de funções globais. Para decidir em qual processo estas informações devem ser armazenadas, uma função de *hashing* é aplicada ao nome global da função.
- Y_GLOBEVAL: a função de *hashing* é aplicada ao nome global fornecido na operação, produzindo a identificação de um processo que mantém as informações necessárias à execução da função global (identificação do processo que a implementa e seu endereço neste processo). Uma mensagem de requisição é então enviada a este processo intermediário e, caso o nome global já tenha sido registrado, a requisição é enviada ao processo que implementa a função. Este processo, então, dispara uma *thread* para executar a função solicitada. Se a função ainda não foi registrada, isto

é, nenhum processo executou Y_GLOBAL, a requisição fica pendente no processo intermediário até que isto ocorra.

- Y_REDUCE: uma função de *hashing* é usada para descobrir qual processo mantém tuplas que satisfazem o *template* especificado. Uma requisição Y_REDUCE é enviada a este processo, contendo o número de tuplas que devem ser removidas, além das operações a serem aplicadas nos campos das tuplas coletadas. Caso não existam tuplas suficientes, pendências são registradas pelo processo (tantas quantas forem as tuplas faltantes). Quando todas as tuplas estiverem disponíveis, as operações de combinação de campos são aplicadas, e a tupla resultante é enviada ao processo que executou Y_REDUCE.
- Y_BARRIER: esta primitiva também utiliza uma função de *hashing* para determinar o processo que deve gerenciar a barreira especificada. Uma requisição Y_BARRIER é enviada a este processo, que pode tratá-la de duas maneiras. Caso seja a primeira requisição para uma determinada barreira, um contador é inicializado com o número de processos que devem se sincronizar. Caso contrário, este contador é decrementado e, quando for igual a zero, o processo que gerencia a barreira envia mensagens a todos os processos bloqueados, liberando-os para execução.

4.3 Inicialização e Término de Aplicações

Antes que os processos de uma aplicação YALI possam se comunicar através do espaço de tuplas, é necessária a execução de um procedimento de inicialização, cujas etapas são descritas a seguir.

- Inicialmente, YaliStart dispara os processos da aplicação, enviando sua própria identificação como um parâmetro para estes processos;
- cada processo cria seus canais (*sockets*) de comunicação, e envia as respectivas identificações a YaliStart;
- YaliStart reúne identificações de canais de todos os processos, enviando uma lista com estas informações a cada processo da aplicação;
- cada processo, após receber esta lista, cria as *threads* necessárias para implementação do espaço de tuplas;
- por fim, é criada uma *thread* para executar a função principal de cada processo, que deve ter sido codificada pelo usuário com o nome YALI_MAIN.

As primeiras três etapas do procedimento de inicialização são necessárias porque os canais de comunicação são criados dinamicamente, e suas identificações não são conhecidas até o momento da execução. O processamento da função principal através de uma *thread* tem o propósito de garantir que, mesmo após seu término, as *threads* que implementam o espaço de tuplas ainda permaneçam executando. Quando a função principal termina, YaliStart é notificado e, quando todos os processos encerram suas funções principais, a aplicação YALI é finalmente encerrada.

5 Avaliação do Ambiente

O propósito desta seção é fornecer uma avaliação preliminar do ambiente YALI, tanto em termos qualitativos como quantitativos. Para permitir uma avaliação geral das características do ambiente, foi organizada uma tabela comparativa entre YALI e algumas implementações Linda citadas na seção 2.1. Em complemento às informações contidas na tabela 1, deve-se mencionar que as operações globais suportadas por p4-Linda são, na verdade, herdadas da biblioteca p4[BOY87], na qual está baseada esta implementação. Também deve-se observar que C-Linda, ao contrário das outras implementações, é um produto comercial, que inclui não somente um pré-processador, mas também um compilador que permite várias otimizações na implementação do modelo Linda. Entre as implementações comparadas, a maioria delas emprega um processo intermediário para gerenciamento do espaço de tuplas, com exceção de YALI e C-Linda.

Tabela 1: Comparação entre YALI e outras implementações Linda.

Sistemas	Processo Gerenciador do TS	Pré-Processador	Política de Distribuição	Operações Globais
C-Linda	não	sim	*hashing*	não
p4-Linda	sim	não	centralizada	sim
Glenda	sim	sim	centralizada	não
POSYBL	sim	não	uniforme	não
YALI	não	sim	*hashing*	sim

Para uma avaliação quantitativa do ambiente foi implementado uma aplicação conhecida como *ping/pong*, onde um processo deposita uma tupla "ping" e espera por uma tupla "pong", enquanto outro retira a tupla "ping" e insere a tupla "pong". Um total de 100 iterações *ping/pong* foram executadas, com tuplas de diferentes tamanhos, fornecendo os resultados contidos na tabela 2. Observando-se esta tabela, nota-se um aumento significativo do tempo para tuplas com mais de 1000 bytes, o que pode ser explicado pela necessidade do estabelecimento de uma conexão quando as tuplas excedem 2Kbytes. Os dados contidos na tabela 2 foram obtidos numa rede de estações Sun, modelo SPARCClassic, num período de baixa utilização da rede.

6 Conclusão e Trabalhos Futuros

Este artigo apresentou a interface e implementação de um ambiente para programação paralela em redes heterogêneas, baseado no modelo Linda. Entre as principais características deste ambiente está o suporte a operações globais, que permitem a comunicação e a sincronização entre um grupo de processos. Outra característica importante consiste na implementação baseada em *threads*, que permite embutir a funcionalidade do espaço de tuplas em cada processo de uma aplicação paralela, diminuindo o volume de comunicação entre processos.

Tabela 2: Tempos médios de cada sequência *ping/pong*.

Tamanho da Tupla (em bytes)	**Tempo Médio** (em milissegundos)
100	9.6
400	10.8
1000	13.3
4000	23.2
10000	57.6

O projeto do ambiente teve uma preocupação muito forte em oferecer uma interface simples e flexível, ao mesmo tempo permitindo a execução eficiente de aplicações paralelas. Mesmo assim, várias otimizações têm sido planejadas. Entre elas estão o desenvolvimento de uma interface gráfica para a configuração das aplicações e a utilização de informações sobre a carga das máquinas para auxiliar na distribuição eficiente dos processos durante a inicialização das aplicações.

A implementação do ambiente ainda está em evolução e, atualmente, está disponível em duas plataformas: SPARC/Solaris e i386/Mach. Está previsto o porte para outras arquiteturas e sistemas que suportam *threads*, como o sistema UNICOS que executa no Cray Y-MP.

Referências

[BOY87] BOYLE, J. et al. *Portable Programs for Parallel Processors*. Hold, Rinehart, and Winston, 1987.

[BUT93] BUTLER, R.M.; LUSK, E. *p4-Linda: A Portable Implementation of Linda*, in Proc. 2nd Int. Symposium on High-Performance Distributing Computing, IEEE Computer Society Press, 1993.

[CAR86] CARRIERO, N.; GELERNTER, D. *The S/Net's Linda Kernel*. ACM Trans. on Computer Systems. New York, v.4, n.2, p.110-129, May 1986.

[CAR94] CARRIERO, N. et al. *The Linda alternative do message-passing systems*. Parallel Computing, n.20, 1994.

[FOS95] FOSTER, I. *Designing and Building Parallel Programs*. Addison Wesley, 1995.

[GEL85] GELERNTER, D. *Generative Communication in Linda*. ACM Trans. on Programming Languages and Systems, v.7, n.1, p.80-112, Jan. 1985.

[NAR89] NAREM Jr. J.E. *An Informal Operational Semantics of C-Linda V2.3.5*. Technical Report, Yale University, Dec. 1989.

[PIN91] PINAKIS, J. *The Design and Implementation of a Distributed Linda Tuple Space*, in Proc. of the Dept. of Computer Science Research Conference, University of Western Australia, 1991.

[SCH91] SCHOINAS, G. *Issues on the implementation of PrOgramming SYstem for distriButed appLications*. Department of Computer Science, University of Crete, Greece, 1991.

[SEY93] SEYFARTH, B. et al. *Glenda Installation and Use*. University of Southern Mississippi, 1993.

DESARROLLO DE UN SISTEMA DE SIMULACIÓN Y PROGRAMACION PARA EL ROBOT EXPERIMENTAL *SALVIATI*

Rafael Fematt M. *Eliseo Juárez R.* *Alfonso Pámanes G.* *Guillermo Franco Q.*

Instituto Tecnológico de la Laguna
Grupo de Robótica del Departamento de Ingeniería Eléctrica y Electrónica
Blvd. Revolución y Cuauhtémoc; Torreón, Coah. 27000; MÉXICO
Dir. electrónica: **apamanes@halcon.laguna.ual.mx** Fax: +52 (17) 13.09.70

RESUMEN

En este artículo se describe el sistema SISPROS, de simulación y programación de tareas del robot manipulador SALVIATI, modelo TL2, desarrollado en el Instituto Tecnológico de la Laguna para fines experimentales y didácticos. Gracias a su simulador gráfico, el SISPROS facilita la programación del robot fuera de línea, lo cual permite planificar un desempeño óptimo de éste para la ejecución de sus tareas. Asimismo es posible realizar la programación del robot en línea. El sistema opera en una computadora personal (PC) en ambiente Windows, y presenta al usuario una interfase gráfica amigable, provista de diversas funciones que simplifican la programación del robot.

1. INTRODUCCIÓN.

Una de las características más interesantes de los robots manipuladores desde el punto de vista industrial es su flexibilidad operacional. En efecto, estas máquinas son capaces de desarrollar múltiples tareas, y cada una de ellas la pueden realizar de diferentes maneras, de tal suerte que se adaptan en poco tiempo y a un costo reducido a las necesidades del usuario.

El proceso de adaptación de un robot a una nueva tarea consiste básicamente en la programación de la secuencia de los nuevos movimientos que deberá ejecutar. Típicamente, la programación de un robot manipulador se realiza bajo dos enfoques: *en línea* y *fuera de línea*. El primer enfoque implica la *enseñanza* de la tarea al robot por parte del usuario, conduciéndolo de tal manera que el órgano terminal se sitúe en un cierto número de puntos correspondientes a la trayectoria a seguir. La programación fuera de línea, en cambio, utiliza un simulador gráfico basado en el modelo matemático del robot y de su ambiente con el propósito de planificar los movimientos del mismo.

Al realizar la programación en línea se requiere la utilización del robot, por lo que éste se vuelve improductivo durante el tiempo necesario para la programación. Ese periodo puede ser bastante prolongado dependiendo de la complejidad de la tarea. No obstante, a favor de este enfoque se tiene el hecho de que el método es relativamente fácil de aprender, lo cual implica una capacitación reducida del programador.

En el método fuera de línea el tiempo de programación se reduce sustancialmente, y la participación del robot es requerida durante un periodo más corto. Otra ventaja importante de

este enfoque es que permite planificar las tareas del robot de tal manera que su desempeño sea óptimo durante la ejecución de éstas (**Pámanes y Zeghloul, 1991**). El procedimiento fuera de línea, sin embargo, implica un costo adicional respecto a la programación en línea debido a la necesidad del uso de herramientas computacionales (*hardware* y *software*) y de instrumentos de calibración.

En la literatura técnica se ha tratado ampliamente el tema de la programación de robots bajo los enfoques aquí mencionados; v.gr. **Derby (1983), Zeghloul et al. (1987), Gini (1987), Buchal et al. (1989).**

En el desarrollo del robot manipulador experimental *SALVIATI*, modelo *TL-2*, se diseñó y realizó un sistema computacional que permite efectuar la programación de tareas bajo cualquiera de los dos enfoques citados en los párrafos precedentes. Este sistema, llamado **SISPROS** (**SI**stema de **S**imulación y **P**rogramación del **RO**bot Salviati), que opera en una computadora personal (PC), está orientado hacia la enseñanza y el entrenamiento para la programación de robots a nivel elemental. La estructura y las funciones básicas del SISPROS se describen en el presente artículo.

2. EL ROBOT MANIPULADOR *SALVIATI TL-2.*

El modelo TL-2 del *SALVIATI* se construyó en el año de 1994, y constituye la segunda versión de este robot experimental y didáctico del Instituto Tecnológico de la Laguna (la primera versión -el modelo TL-1- se realizó en 1987). Como se indica en el esquema de la Figura 1, el robot está integrado por los siguientes sistemas: mecánico, de control, de comunicación, y de percepción.

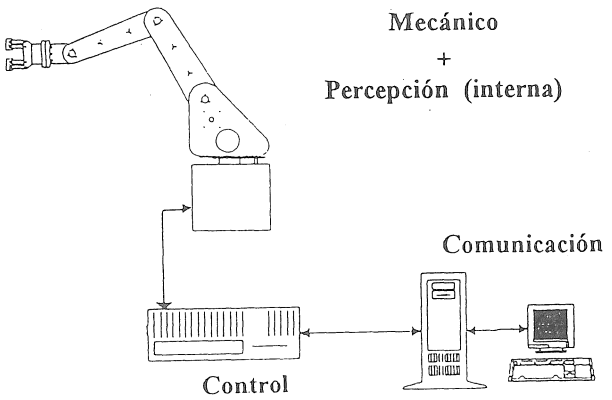


Figura 1. Sistemas del robot manipulador *SALVIATI.*

El sistema mecánico consiste en un manipulador de cuatro grados de libertad, los actuadores y las transmisiones de potencia. El manipulador posee una arquitectura de coordenadas angulares, como se muestra en la Figura 2, en donde también se representa la sección transversal del espacio de trabajo. Los motores son eléctricos de corriente directa, a partir de los cuales se conduce la potencia hacia los eslabones del manipulador mediante transmisiones de engranes, cadenas y bandas dentadas (**Pámanes** et al, **1995**).

El sistema de percepción está constituido por cuatro captadores analógicos, instalados en las articulaciones del manipulador, los cuales miden las coordenadas articulares.

La tarjeta principal del sistema de control del *SALVIATI* se basa en el uso de microcontroladores HCTL-1000 de *Hewlett Packard.* Esta tarjeta es la misma que se diseñó para el modelo TL-1, y fue desarrollada en cooperación con el Instituto de Automática Industrial (IAI), de Madrid, España, en el marco del programa CYTED (**Santibáñez** et al, 1990). Además de esta tarjeta, en el sistema de control se incluye una interfase de potencia, que amplifica las señales de control para alimentar a los actuadores.

Finalmente, el sistema de comunicación incluye una computadora PC 486 DX2, y al paquete **SISPROS** para la programación y simulación del robot.

Las características de operación del *SALVIATI TL2* son las siguientes:

- Capacidad de carga: **9.81 N**
- Alcance máximo: **0.90 m**
- Velocidad máxima: **0.50 m/s**

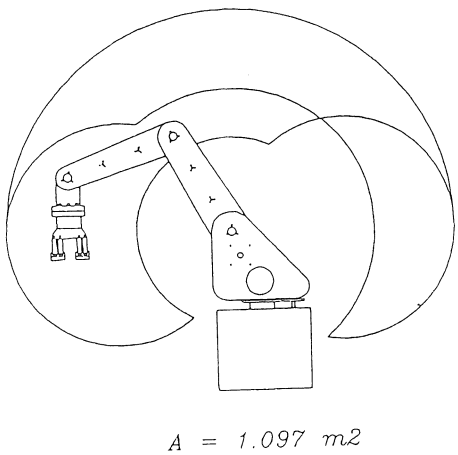


Figura 2. Sección transversal del espacio de trabajo del robot *SALVIATI*

3. DESCRIPCIÓN GENERAL DEL *SISPROS.*

La misión del programa **SISPROS** consiste en establecer la comunicación entre el usuario y los otros sistemas del robot. Para ello el programa posee funciones que ofrecen al usuario las siguientes posibilidades:

- Informarse acerca de la situación del manipulador en términos de sus coordenadas operacionales y sus coordenadas articulares.
- Asignarle al manipulador los programas de movimientos necesarios para la realización de sus tareas.
- Modificar los valores de las ganancias para las leyes de control de los motores.
- Utilizar el recurso de la simulación gráfica para la planificación y evaluación de los movimientos del manipulador antes de su ejecución en tiempo real.

La *ventana* inicial del programa que se le presenta al usuario está organizada en cuatro secciones (ver la Figura 3). La ***sección de animación*** corresponde al área de la pantalla reservada para la simulación gráfica del manipulador. En la ***sección de comando articular*** se presenta una *botonadura* para la generación de desplazamientos articulares, controlados por el usuario. La ***sección de programación*** contiene un conjunto de *íconos* asociados a las funciones necesarias para el establecimiento de un programa de movimientos. Finalmente la ***sección de coordenadas articulares*** presenta cuatro pequeñas ventanas que exhiben los valores de dichas coordenadas. En la Tabla 1 se presenta un resumen de las funciones disponibles en cada una de estas secciones.

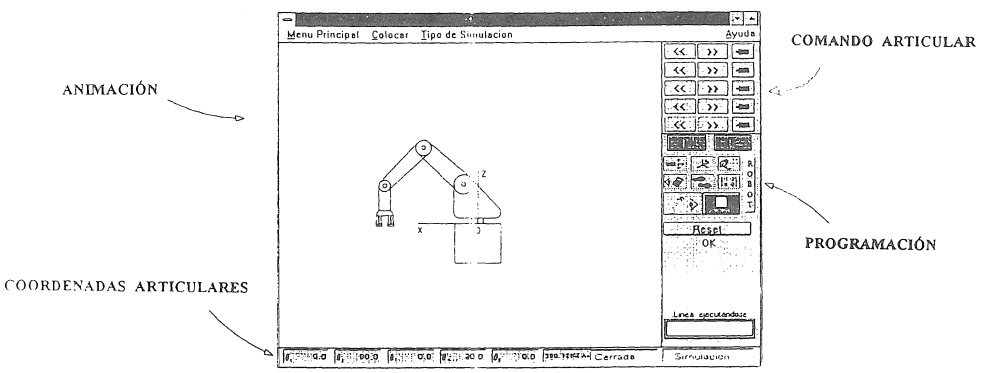


Figura 3. Ventana principal del SISPROS.

Tabla 1. Resumen de funciones del paquete *SISPROS.*

FUNCIONES DE CADA SECCIÓN	*ÍCONO O BOTÓN*	*OPCIONES*
SECCION ANIMACIÓN		
* Simulación.		
* Menú de entrada:		- Gestión de archivos, salir del sistema. - Restablecimiento de la configuración del manipulador. - Modo de operación (*en línea* y *fuera de línea*).
SECCION COMANDO ARTICULAR		
* Desplazamiento articular:		- Selección de la articulación (1, 2, 3, 4).
* Modificación de parámetros de ley de control:		- Selección del motor (1, 2, 3, 4).
SECCION PROGRAMACIÓN		
* Comando de la pinza:		- Apertura y cierre.
* Desplazamiento ortogonal de la pinza:		- Dirección de desplazamiento (X, Y, Z).
* Punto de observación de la escena simulada:		- Selección de cualquier punto de observación.
* Grabación de configuraciones:		- Grabar una configuración presentada en pantalla. - Suprimir una configuración existente en archivo. - Ejecutar una configuración existente en archivo.
* Exhibición de coordenadas operacionales:		- Coordenadas cartesianas de la pinza. - Ángulos de Euler. - Matriz homogénea de la pinza.
* Modificación del *paso*:		- Paso de las rotaciones articulares. - Paso de desplazamiento lineal de la pinza.
* Consigna de coordenadas operacionales:		- Especificar la matriz homogénea deseada de la pinza.
* Observación de movimientos en línea:		- Abrir y cerrar la ventana de observación.
* Edición de programas:		- Edición programas de movimientos. - Gestión de archivos de programas. - Ejecución y paro de programas.
SECCION COORDENADAS ARTICULARES		
* Exhibición de las coordenadas articulares.		

4. PROCESO DE PROGRAMACIÓN.

Para facilitar la enseñanza, el SISPROS cuenta con una botonadura gráfica equivalente a un *teach pendant* industrial que le permite al usuario, con la ayuda del *ratón*, controlar los desplazamientos de la pinza en las direcciones de los ejes de un marco ortogonal fijo. La implantación de esta función se basa en el uso de los modelos cinemáticos de posición del manipulador, como se ilustra en el esquema de la Figura 4. En el procedimiento ahí mostrado, la situación de la pinza es especificada implícitamente por el usuario al demandar la ejecución de un desplazamiento en cualquier dirección, considerando el punto de partida y el valor vigente del paso. Estos desplazamientos se pueden realizar ***en línea*** o ***fuera de línea***. En ambos casos se simulan los movimientos del manipulador en la sección de animación. Durante los desplazamientos de la pinza su orientación se conserva. La orientación puede ser establecida por el usuario activando los botones para la rotación de la cuarta articulación (sección de comando articular).

Una vez que la pinza se encuentra en la situación deseada, la configuración del manipulador se registra en un archivo (Figura 5). A cada configuración registrada se le asigna un número consecutivo, con el que se identificará posteriormente durante la edición del programa de movimientos. En la ventana de edición se prescribe la secuencia deseada de ejecución de configuraciones previamente registradas, y se asignan instrucciones suplementarias tales como ***pausa***, ***saltar a...*** ,y ***abrir / cerrar pinza***. Adicionalmente, desde esta ventana se ordena la ejecución / paro del programa, y la suspensión temporal .

El procedimiento descrito en los párrafos precedentes se puede seguir también en la programación ***fuera de línea***. En este caso no es necesaria la participación del manipulador. Los movimientos de éste se simulan gráficamente para guiar al usuario en la definición de las configuraciones requeridas por la tarea. Para definir el modo de programación es necesario hacer la selección pertinente en el menú de la sección de animación.

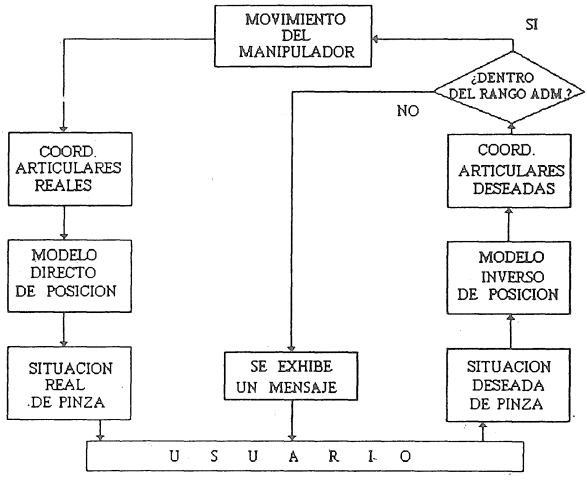


Figura 4. Estrategia para los desplazamientos ortogonales de la pinza

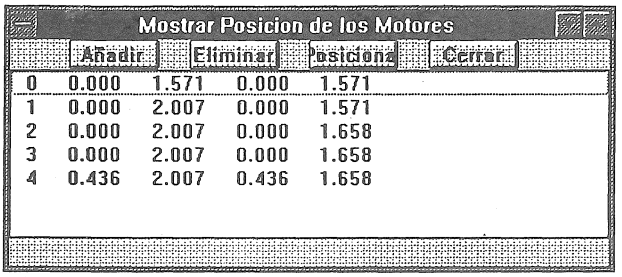


Figura 5. Ventana para la grabación de configuraciones.

En la Figura 6 se muestra la secuencia de configuraciones que se observa en el monitor de la computadora durante la simulación de la tarea definida por las tres posiciones especificadas en la Tabla 2. La orientación de la pinza es tal que debe mantenerse en dirección vertical, con los dedos apuntando hacia abajo, en los tres puntos. Esta tarea fue programada en el modo ***fuera de línea***, desplazando la pinza de manera interactiva con la ayuda de la función de ***desplazamiento ortogonal***. La orientación fue establecida utilizando los botones de la cuarta articulación, de la sección de comando articular. Las coordenadas articulares que se dan en la Tabla 2 son obtenidas automáticamente en el **SISPROS** al colocar la pinza en las sucesivas posiciones de la tarea.

5. CONCLUSIÓN.

En el presente trabajo se describieron las características básicas del paquete computacional **SISPROS**, utilizado para la simulación y programación del robot manipulador *SALVIATI TL-2*, desarrollado en el Insitituto Tecnológico de la Laguna. El **SISPROS** está orientado hacia la enseñanza y el entrenamiento para la programación de robots. Su utilización es muy simple gracias al uso de una interfase gráfica en ambiente *Windows*, y a la simulación gráfica de los movimientos del manipulador. Esta última característica permite realizar la programación del robot ***fuera de línea***.

Actualmente, el sistema de control del robot solo cuenta con un lazo de control de posición para cada actuador, y no dispone de la etapa de velocidad. Por ello, en la versión actual del **SISPROS** no se incluye la opción de programación de velocidad. En el desarrollo futuro del robot se prevé la implantación del control y la programación de velocidad. En el **SISPROS**, adicionalmente, se contempla la incorporación de diversas funciones orientadas al desarrollo de la parte de programación ***fuera de línea***. Actualmente se trabaja en la implantación de un procedimiento interactivo para la determinación del emplazamiento óptimo del manipulador en relación a su tarea. Otro trabajo en curso se refiere a la incorporación de un algoritmo para la generación automática de trayectorias teniendo en cuenta la prevención de colisiones.

Tabla 2. Posición de la pinza y coordenadas articulares del manipulador. Tarea de la Figura 6.

Punto	*X* *(cm)*	*Y* *(cm)*	*Z* *(cm)*	θ_1 (°)	θ_2 (°)	θ_3 (°)	θ_4 (°)
P1	25	0	18	0	60.3	-41.0	-109.3
P2	25	0	8	0	59.6	-80.6	-69.0
P3	25	0	0	0	42.5	-87.6	-44.9

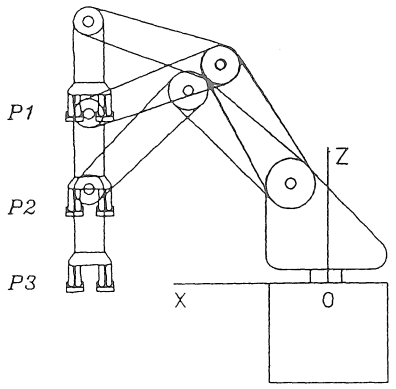


Figura 6. Simulación de una tarea en el SISPROS.

REFERENCIAS BIBLIOGRÁFICAS.

Buchal R.O., Chercas D.B., Sassani F., Duncan J.P.; *Simulated off-line programming of welding robots*; International Journal of Robotics Research, Vol. 8, No. 3, pp. 31-43 (1989).

Derby S.; *Simulating motion elements of general purpose robot arms*; International Journal of Robotics Research, Vol. 2, No. 1, pp. 3-12 (1983).

Fuentes O., Santibáñez V., Fematt A., Pámanes A.; *Mejoras a los sistemas mecánico, de control y de comunicación de un prototipo de robot manipulador*; Informe Técnico del Proyecto 918.88 del Centro de Graduados e Investigación del Instituto Tecnológico de la Laguna.

Gini M.; *The future of robot programming*; Robotica, Vol. 5, pp.235-246 (1987).

Pámanes A., Zeghloul S.; *Optimal placement of robotic manipulators using multiple kinematic criteria*; Proceedings of the IEEE International Conference on Robotics and Automation, Vol.1, pp.933-938 (1991).

Pámanes A., Fematt A., Franco G, Juárez E., Guajardo E.; *Mejoras en el diseño del robot manipulador SALVIATI.* Memorias del I Congreso de la Sociedad Mexicana de Ingeniería Mecánica (1995). pp. 157-162

Santibáñez D.V., Armada R.M., González de S.P.; *Red de adelanto para control digital de motores;* Informe Técnico IAI-UEIASS-001/90; Instituto de Automática Industrial; Madrid, España (1990).

Zeghloul S., Murguet D., Lallemand J.P.; *Outils de simulation et de programation hors ligne d'environnements multi-robots*; Proceedings of the 7-th IFToMM Congress on Theory of Machines and Mechanisms, pp. 1089-1092 (1987).

Asistente de planificación de trayectorias libres de colisión

María Alejandra Godoy
Departamento de Ingeniería de Sistemas
Universidad de los Andes
e-mail: mgodoy@uniandes.edu.co

Jorge E. Morales
Facultad de Ingeniería de Sistemas
Fundación Universitaria San Martín
e-mail: jmorales@uniandes.edu.co

Resumen: El objetivo de este artículo es presentar un asistente para un planificador de trayectorias libres de colisión para robots manipuladores. La función del asistente es automatizar la definición de los parámetros de un planificador de trayectorias libres de colisión a partir de un análisis del robot y del entorno que rodea el robot. Aunque el asistente se detalla para planificación en robots manipuladores, puede extenderse para planificación en otros tipos de robots.

Palabras claves: Robótica, inteligencia artificial, planificación de trayectorias, manipuladores.

1. Introducción

El problema de la planificación de trayectorias libres de colisión trata con la generación de caminos para un objeto en movimiento o manipulador desde una configuración inicial a una configuración final sin colisionar con los obstáculos que rodean el manipulador. Con base en pruebas experimentales [1] se ha notado que el desempeño de un planificador de trayectorias depende de información del mundo que rodea el robot como por ejemplo: distribución de los objetos en la escena, claridad entre ellos, geometría, etc. Otra característica de algunos planificadores es que éstos requieren de ciertos parámetros para su ejecución los cuales son definidos por el usuario y, en la mayoría de los casos, éste no puede escoger correctamente un valor que esté de acuerdo con la información de la escena ya que no conoce la relación entre ésta y el parámetro. Estos parámetros, además, son invariantes en tiempo de ejecución del planificador.

La solución que se propone es desarrollar un **asistente de planificación** que, a partir de un análisis de las características de la escena y de la información de comportamiento del planificador, pueda definir automáticamente, sin interacción alguna con el usuario, los valores para los diferentes parámetros del método. Así mismo, el asistente debe servir para lograr que estos parámetros sean dinámicos, es decir, que varíen de acuerdo a la localidad que rodea la configuración de la trayectoria que se está analizando.

A continuación se hace un repaso del planificador de Choque y Arrastre [2] sobre el cual se va a implementar la solución propuesta; en la sección 3 se presenta una descripción formal del problema; en la sección 4 se presenta una visión general de la solución; en la sección 5 se expone el caso particular de un asistente para el planificador mencionado y por último, en la sección 6 se presentan las conclusiones.

2. Método de planificación de Choque y Arrastre

El planificador de Choque y Arrastre [2] intenta viajar hacia la configuración final por la trayectoria de interpolación articular lineal. En el momento en que detecta una colisión con un obstáculo, el robot trata de "arrastrarse" sobre éste buscando enrutarse nuevamente sobre la trayectoria inicial. Este planificador trabaja sobre el EC sin precalcular y genera solamente lo indispensable de éste durante el proceso de búsqueda de soluciones.

La manera como el robot rodea el obstáculo en el EC está guiada por una estrategia de evasión, la cual involucra el desplazamiento de una o dos articulaciones principalmente (Figura 2.1.). El robot continúa guiándose con la estrategia inicialmente escogida hasta cuando logre evitar el obstáculo o hasta que las articulaciones lleguen a sus límites, en cuyo caso debe intentar por otra dirección, es decir, escoger otra estrategia de evasión.

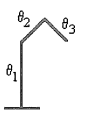


Figura 2.1. Articulaciones involucradas en las diferentes estrategias de evasión

El método está desarrollado para soportar tres tipos de estrategias de evasión: en la estrategia de evasión vertical (Figura 2.2a.) el robot trata de evitar el obstáculo rodeándolo en el EC, por encima y por debajo moviendo θ_3 ó θ_2 y por la izquierda y por la derecha moviendo θ_1. En la estrategia diagonal (Figura 2.2b.), el robot trata de evitar el obstáculo rodeándolo por las diagonales superior izquierda, superior derecha, inferior derecha e inferior izquierda, moviendo las articulacionesθ_1 y θ_2 ó θ_1 y θ_3 ó θ_2 y θ_3. En la estrategia combinada (Figura 2.2c.), el robot intenta rodear el obstáculo tanto por las direcciones de la estrategia vertical como por las de la diagonal.

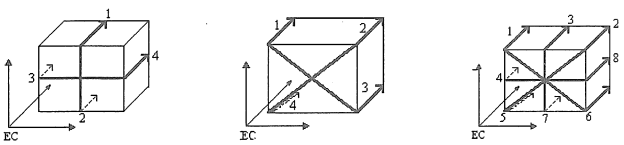


Figura 2.2. (a) Estrategia de evasión vertical en el EC. (b) Estrategia diagonal. (c) Estrategia combinada

Para probar el desempeño del método de Choque y Arrastre se realizó un proceso de experimentación y evaluación [1] el cual fue llevado a cabo en una estación IBM RISC-6000, utilizando MSIM [3]. Durante el proceso de evaluación se identificaron y caracterizaron las tareas que éste resuelve de acuerdo al tiempo de respuesta:

- Las tareas simples son aquellas en las cuales hay pocos obstáculos entre CI y CF, y en donde las necesidades de planificación a nivel de pinza son bajas (tiempo promedio de respuesta: 2 seg).

- Las tareas no triviales son aquellas en las que son necesarias pequeñas maniobras en la mano para encontrar solución y/o cuando aumenta el número de obstáculos que impiden el acercamiento al punto final (tiempo promedio de respuesta: 4 a 5 seg).
- Las tareas más difíciles de resolver se presentan cuando se hacen necesarios movimientos precisos de la mano para encontrar una solución como por ejemplo, una planificación en la que el robot debe insertar un objeto en otro. Para estos casos el tiempo promedio de respuesta moviendo la articulaciónθ_2 es de 9 seg. y moviendo la articulación θ_3 es de 13 seg. El orden en que son aplicadas las estrategias para evadir los obstáculos influye de una forma directa en el costo de la solución y en el tiempo de respuesta.
- Los casos de falla son aquellos en los que la planificación en la pinza es la única forma de sacar al robot de algún hueco. La mayoría de estos casos fallan debido a que el método no está pensado para trabajar en dichas situaciones.

3. Descripción formal de la planificación

Un planificador de trayectorias es una herramienta compuesta por un conjunto de algoritmos especializados en encontrar una trayectoria libre de colisiones para llevar a un manipulador de un lugar a otro en un mundo dado [2] donde una trayectoria libre de colisiones entre CI y CF se define como:

$$TLC = <C_1, ..., C_n> \mid C_1 = CI,\ C_n = CF\ \Lambda \forall i\ \neg\exists \text{ colisión en } C_i, \text{ ni al pasar de } C_i \rightarrow C_{i+1}$$

Un planificador de trayectorias *pt* es una función dependiente de ciertos parámetros de entrada:

$$pt\ (R, M, CI, CF, P) = TLC$$

donde R es el robot, M es el mundo, CI y CF son las configuraciones inicial y final respectivamente y P es el conjunto de valores de los parámetros que necesita el planificador para calcular la trayectoria solución entre CI y CF.

4. La solución propuesta

En la actualidad, un planificador de trayectorias se puede ver como una caja negra a la cual entran los siguientes datos: el robot R, el mundo M, las configuraciones inicial CI y final CF y los parámetros P y de la cual sale una trayectoria libre de colisión TLC de CI a CF:

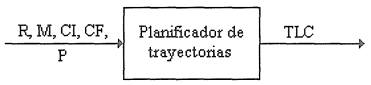


Figura 4.1. Etapas del planificador de trayectorias, actualmente

El asistente de planificación propuesto realizaría dos etapas que buscan mejorar el desempeño del planificador: una de resolución de parámetros y una etapa de retroalimentación. El objetivo de la primera etapa es definir los valores de los parámetros de entrada del planificador a partir de un análisis global de la

información de la escena que puede ser relevante para un buen desempeño del planificador. En las pruebas experimentales realizadas en [1], se puede ver que entre los parámetros del planificador y las características de la escena debe existir alguna función de dependencia que debe ser aprovechada por el asistente de planificación para calcular los valores para cada uno de dichos parámetros, así :

$$P_i = f(C),\ 1 \leq i \leq n$$

donde P_i es un parámetro de planificación y C es el conjunto de características de las escenas.

El objetivo de la etapa de retroalimentación es permitir que los valores de los parámetros del planificador sean dinámicos, ya que el mundo local a lo largo de la trayectoria solución es diferente. En realidad, ésta no es una etapa adicional sino más bien un llamado a la anterior etapa de resolución de parámetros cada vez que sea necesario recalcular los parámetros para hacerlos más acordes con el mundo local.

En la Figura 4.2. se puede ver cómo el asistente entraría a apoyar al planificador: la idea es que el usuario ya no deba definir ningún parámetro; sólo tendría que escoger las configuraciones entre las cuales quiere hallar una trayectoria libre de colisión. Entonces, al asistente deben entrar el robot R, el mundo M y las dos configuraciones CI y CF y con base en el análisis de estos parámetros se pretende definir el conjunto de parámetros P', propios del planificador. Adicionalmente, se desea que cuando el entorno local cambie, cambie también el conjunto P'. Con el fin de lograr este objetivo, se crea una etapa de retroalimentación en la que el planificador se apoyaría en el asistente para que éste defina nuevamente el conjunto P' de acuerdo al nuevo entorno local. Debido a que se haría un análisis local, sería necesario pasarle al asistente la configuración actual C_i del robot.

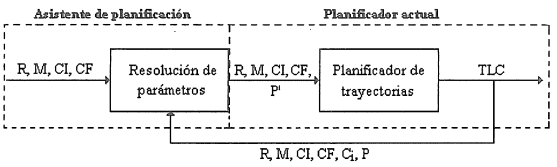


Figura 4.2. Etapas del planificador de trayectorias propuesto

5. Un caso particular

En el desarrollo de esta propuesta se escogió llevarla a cabo sobre el planificador de Choque y Arrastre por las siguientes razones:

- Los resultados de las pruebas realizadas en [1] sugieren la existencia de una posible dependencia entre el desempeño del planificador y características del entorno.
- Actualmente, el orden de aplicación de las estrategias de evasión es siempre el mismo sin importar la geometría o posición del objeto a evadir.

- Es un planificador muy eficiente aún con la desventaja del orden estático de aplicación de las estrategias: sus medidas de respuesta en términos de tiempo de planificación, costo de la solución, número de nodos y número de movimientos, son comparativamente mucho más bajas con respecto a los otros planificadores de MSIM.

5.1. Resolución de parámetros

Recordemos que la función de esta etapa es definir los valores de los parámetros de entrada del planificador de trayectorias para evitar la intervención del usuario y poder calcular unos valores de parámetros lo más acorde posible con la información de la escena.

Para el caso particular del planificador de Choque y Arrastre los parámetros que deben ser resueltos son:

- el grado de la discretización del espacio de configuraciones en el proceso de búsqueda del choque.
- el orden de aplicación de las diferentes direcciones en la estrategia de arrastre.

En la sección 2 se habló de la estrategia de evasión, otro parámetro existente en el planificador actual. Este parámetro fue "eliminado"porque se tomó la decisión de que el asistente analice todas las direcciones de evasión de la estrategia total porque es un proceso generalizable al número de direcciones que se requiera. A continuación se hace describen los parámetros que deben ser resueltos por el asistente, las diferentes alternativas tenidas en cuenta para resolver dichos parámetros y la alternativa escogida.

5.1.1. Discretización del EC en el proceso de búsqueda del choque

Es el nivel de partición que se va a hacer sobre el espacio de configuraciones cuando el robot se encuentra cerca de un obstáculo, es decir, de una posible colisión. Esto significa que cuando no se está cerca de un objeto, el robot puede moverse con libertad pero cuando éste se encuentra cerca de algún obstáculo, debe tener más cuidado en los movimientos que realiza debido a la poca libertad que tiene para moverse. Entonces, para definir este parámetro debe analizar **el nivel de libertad de movimiento** que puede ser medido con diferentes criterios:

a) Conos generalizados en el espacio físico: es un modelo de representación del espacio libre en el espacio físico, por medio de un conjunto de conos formados a partir del barrido y deformación de una sección unidimensional a lo largo de una curva en el espacio llamada columna o espina [4]. La idea es conectar los diferentes conos formando un grafo por el cual viaja el robot (Figura 5.2a.)

b) Zonas de barrido en el espacio físico: son los sectores circulares en los que cada una de las articulaciones se puede mover de su posición actual sin chocar con un objeto en el espacio físico. Esta medida está representada por una n-tupla de la siguiente forma

$$Z = (\theta_{1p}, \theta_{1n}, ..., \theta_{Np}, \theta_{Nn})$$

donde N es el número de grados de libertad del robot, θ_{ip} y θ_{in} indican que el ángulo es en la dirección positiva o negativa de la articulación, respectivamente, tal como lo muestra la Figura 5.2b.

c) Claridad en el espacio de configuraciones: esta medida se define como la mínima distancia desde la configuración actual hasta los vértices de los obstáculos en el espacio de configuraciones. Esta alternativa fue la que se implementó en el asistente de planificación (Figura 5.2c.)

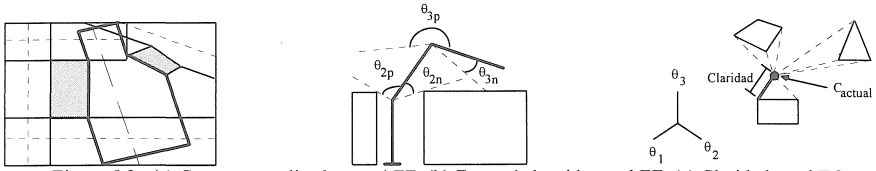


Figura 5.2. (a) Conos generalizados en el EF. (b) Zonas de barrido en el EF. (c) Claridad en el EC.

5.1.2. Orden de aplicación de las direcciones de evasión

El orden de aplicación de las direcciones de evasión indica la dirección hacia la cual debe moverse cada articulación y debe ser una función del **estado actual del mundo local** que es resultado del estado de vecindad del objeto de choque, del estado del objeto de choque y del estado del robot en el momento del choque.

5.1.2.1. Estado de vecindad del objeto de choque

El estado de la vecindad del objeto de choque da información acerca de los objetos que rodean este obstáculo. Esta medida puede ser obtenida con criterios como:

a) Densidad alrededor del objeto: con esta medida se conoce la densidad de las n zonas que rodean el obstáculo de choque donde n es el número de direcciones que se van a tener en cuenta para rodear el obstáculo. Las diferentes direcciones se ordenan de acuerdo a qué tan densas están sus respectivas zonas.

b) Tipo de objetos: es una clasificación de los objetos que rodean el actual y da información sobre el aspecto externo de un objeto: qué tan feo es, si tiene huecos, etc. Esta clasificación ayuda a que el robot evite rodear estos objetos ya que hacen más difícil el arrastre en caso de un posible choque.

c) Número de obstáculos alrededor del objeto: con esta medida se conoce la distribución de las n zonas que rodean el objeto de choque. Esta alternativa fue la que se implementó en el asistente de planificación.

Los aspectos de comparación de las diferentes alternativas son la capacidad de medida, es decir, qué tanto dice dicha medida acerca de la vecindad del objeto de choque y la facilidad de implementación de cálculo.

	Capacidad de medida	Implementación
Densidad	✓	½
No. de objetos	½	✓
Tipo de objetos	½	✗

Tabla 5.1. Comparación de medidas del estado de la vecindad del objeto de choque

5.1.2.2. Estado del objeto de choque

El estado del objeto de choque da información del objeto con el cual se colisiona el robot. Esta medida puede ser obtenida con criterios como los que se exponen a continuación pero para el asistente no se tomó en cuenta ninguna por estar fuera del alcance del desarrollo.

a) Distancia del punto de choque a los vértices y las aristas de la cara de choque: esta medida tiene la desventaja de que es incompleta ya que esta distancia es muy local y si tomamos la dirección hacia la cual obtuvimos la menor distancia, podemos encontrar que el objeto hacia esa dirección es más largo o complicado de recorrer.

b) Distancia del punto de choque a los vértices y las aristas del minimax del objeto proyectado en el plano de la cara de choque: esta medida es completa ya que se está tomando en cuenta todo el objeto de choque.

c) Orientación del objeto de choque: esta medida indica si el objeto está acostado o parado (es decir, es más ancho que largo o viceversa).

	Capacidad de medida	Implementación
Distancias en la cara de choque	✗	✓
Distancias en el minimax de la cara de choque	✓	✓
Orientación del objeto	✓	✗

Tabla 5.2. Comparación de las medidas del estado del objeto de choque

5.1.2.3. Estado del robot en el momento del choque

Esta medida nos indica cómo se encuentra el robot en el momento de colisionar con un objeto y se puede obtener con los siguientes criterios:

a) Posición del robot en el EC: es la configuración actual en la que se encuentran las articulaciones del robot. Esta información sirve para saber si el robot se encuentra cerca de los límites de sus articulaciones.

b) Tipo de configuración: indica en qué configuración se encuentra el robot. Por ejemplo, para un PUMA560 indica si su hombro está en configuración derecha o izquierda, si el codo está arriba o abajo y si la muñeca está arriba o abajo.

c) Rango de movilidad: es la medida de cuanto le falta a cada articulación para llegar a sus límites mínimo y máximo. Esta medida sirve para saber si el robot se encuentra cerca de los límites de sus articulaciones y ayuda a evitar tomar una dirección de evasión que lo acerque más a dichos límites.

	Capacidad de medida	Implementación
Posición del robot en el EC	½	✓
Rango de movilidad	½	✓
Tipo de configuración	✗	✓

Tabla 5.3. Comparación de las medidas del estado del robot en el momento del choque

5.1.2.4. Cálculo del número de objetos alrededor del objeto de choque

Para encontrar el número de objetos que rodea el objeto de choque, se proyecta el centroide de cada obstáculo en el plano perpendicular a la dirección de la articulación θ_3 (Figura 5.5a.) de la siguiente forma: se calcula primero el sistema de referencia de la articulación, se trasladan los centroides de cada objeto a este nuevo sistema y se proyecta el punto al plano xy, es decir, se elimina la coordenada z del punto (Figura 5.5b.). Una vez se ha proyectado el punto, se analiza a qué zona pertenece y se incrementa el número de objetos de dicha zona (Figura 5.5c.).

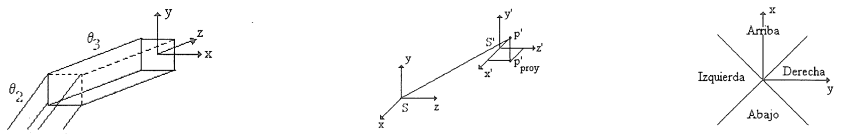


Figura 5.5. (a) Sistema de referencia de la articulación θ_3. (b) Traslación y proyección del centroide a S'. (c) Zonas de análisis para la estrategia vertical

5.1.2.5. Cálculo del rango de movilidad del robot

Esta medida está dada por la siguiente definición

$$RM+ = |\ Cactual<i> - Cmax<i>\ |,\quad \text{para } 1 \le i \le 3$$

$$RM- = |\ Cactual<i> - Cmin<i>\ |,\quad \text{para } 1 \le i \le 3$$

donde RM+ y RM- son los rangos de movilidad en la dirección positiva y negativa de la articulación, respectivamente.

De los resultados de los dos anteriores cálculos se obtiene el orden de las direcciones de evasión así: los resultados del cálculo del número de objetos son ordenados de menor a mayor (se le da prioridad a una dirección con un número de objetos mínimo) y los resultados del rango de movilidad son ordenados de mayor a menor (se le da prioridad a una dirección con mayor movilidad). Finalmente, se mezclan las dos listas ordenadas para obtener el orden final de las direcciones de evasión:

$$O = peso1*estado_vecindad_{dirección} + peso2*estado_robot_{dirección}$$

Para asignar peso1 y peso2 se revisa si existe alguna articulación cuyo rango de movilidad sea más bajo que un epsilon dado. Si este es el caso se le da más peso al estado del robot con el fin de tratar de evitar que el robot tome una dirección en la cual no pueda moverse más. De lo contrario, se le da un mayor peso a la vecindad para tratar de evitar colisiones futuras con otros objetos que rodean el objeto de choque.

5.2. Resultados experimentales

Todas las pruebas realizadas para el método de Choque y Arrastre sin asistente, se realizaron con un valor de un grado para la discretización del espacio de configuraciones debido a que, con este valor, el método siempre encuentra solución si existe una ruta desde la configuración inicial hasta la final. Así mismo, es

importante anotar que se tomó la misma pareja de configuraciones para cada serie de pruebas logrando que los resultados no fueran dependientes de la ruta escogida. Las abreviaturas de las tablas de desempeño son las siguientes:

T: Tiempo de planificación (A: asistente y C: choque)

N: Número de nodos en la solución

C: Costo en grados de la solución

M: Número de movimientos de la solución

I: Número de intentos fallidos antes de encontrar la dirección de evasión final

Por último, los puntos oscuros en los gráficos pertenecen a los resultados del asistente y los puntos claros pertenecen a Choque y Arrastre.

Los resultados obtenidos para la prueba de Claridad vs. Desempeño se lograron a partir de escenas con dos objetos en las que se varió la distancia de separación entre ellos.

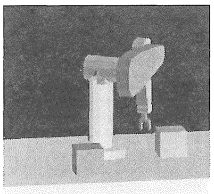


Figura 5.6. Tipo de escena para medir Claridad vs. Desempeño

Tipo de prueba	TA	TC	NA	NC	CA	CC	MA	MC
Muy cerca	0.55	0.69	101	235	231.40	234.41	103	236
Cerca	0.67	0.74	131	252	252.82	251.83	134	254
Lejos	0.75	0.78	151	250	256.83	249.83	153	252
Muy Lejos	0.94	0.82	180	266	306.06	265.41	186	268

Tabla 5.1. Medidas para Claridad vs. Desempeño

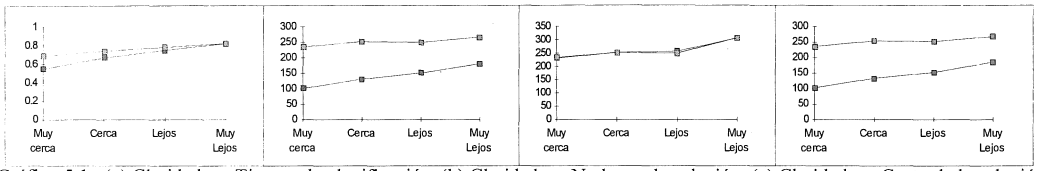


Gráfico 5.1. (a) Claridad vs. Tiempo de planificación. (b) Claridad vs. Nodos en la solución. (c) Claridad vs. Costo de la solución. (d) Claridad vs. Número de movimientos

Para la prueba de Distribución de los obstáculos alrededor del objeto de choque vs. Desempeño se utilizaron escenas en las que el robot sólo colisiona con un objeto y alrededor de éste hay un grupo de objetos que en cada escena está en una posición distinta.

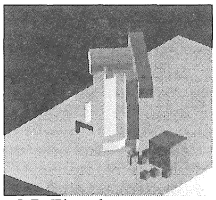


Figura 5.7. Tipo de escena para medir Distribución vs. Desempeño

Tipo de prueba	TA	TC	NA	NC	CA	CC	MA	MC	IA	IC
Arriba	0.48	0.64	71.0	89.5	86.41	88.42	71.0	90.5	0	1
Derecha	0.50	0.63	80.0	96.5	93.41	95.42	80.0	97.5	0	1
Abajo	0.49	0.81	86.5	151.0	104.41	153.7	86.5	152.0	0	1
Izquierda	0.55	0.64	71.5	89.5	87.39	88.42	71.5	90.5	0	1

Tabla 5.2. Medidas para Distribución vs. Desempeño

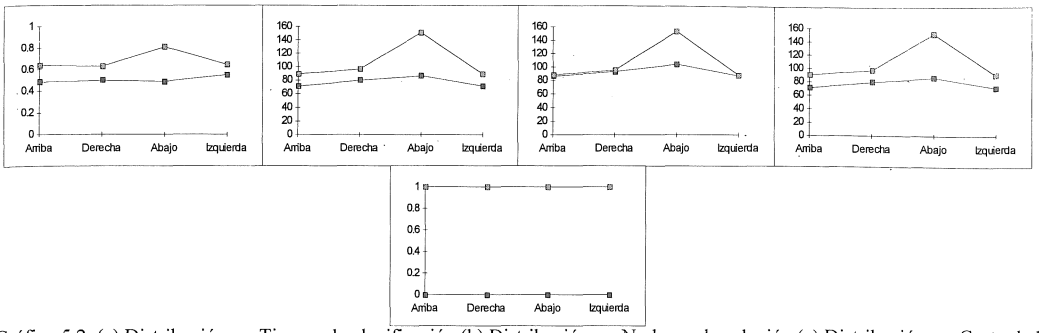


Gráfico 5.2. (a) Distribución vs. Tiempo de planificación (b) Distribución vs. Nodos en la solución (c) Distribución vs. Costo de la solución. (d) Distribución vs. Número de movimientos (e) Distribución vs. Número de intentos fallidos

Para la prueba de Rango de movilidad del robot vs. Desempeño se utilizaron escenas en las que alguna de las tres articulaciones de posicionamiento del robot se encuentra cerca de sus límites en el momento de la colisión, obligando al robot a que no evada el objeto en esa dirección.

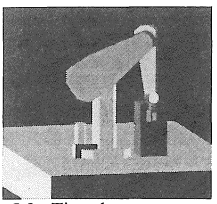


Figura 5.8. Tipo de escena para medir Movilidad vs. Desempeño

Prueba	TA	TC	NA	NC	CA	CC	MA	MC	IA	IC
1	0.54	1.95	91	147	147.37	146.36	91	148	0	1
2	0.67	2.01	101	153	163.49	162.50	101	153	0	2
3	0.55	1.97	98	142	145.10	147.38	99	144	0	1
4	0.72	2.11	151	164	183.74	185.23	153	164	0	2

Tabla 5.3. Medidas para Movilidad vs. Desempeño

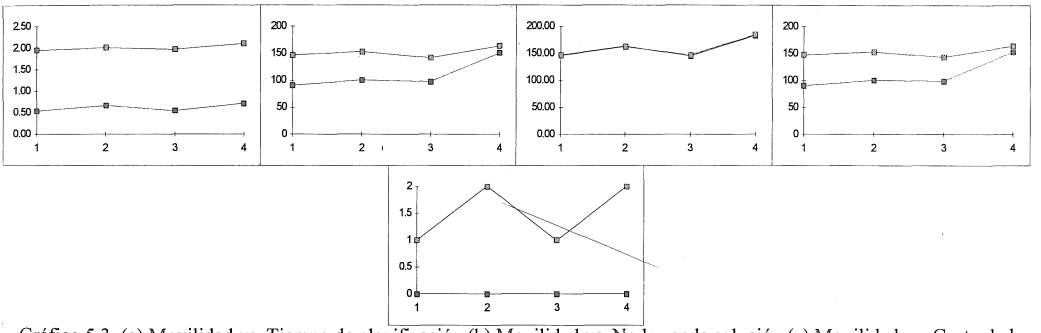


Gráfico 5.3. (a) Movilidad vs. Tiempo de planificación (b) Movilidad vs. Nodos en la solución (c) Movilidad vs. Costo de la solución (d) Movilidad vs. Número de movimientos (e) Movilidad vs. Número de intentos fallidos

En la primera prueba, el tiempo de planificación mejora al tener en cuenta la claridad, ya que el robot avanza con movimientos más largos cuando tiene libertad para moverse. Si analizamos el gráfico vemos que la diferencia entre los tiempos es mayor cuando los objetos están cerca que cuando la distancia de separación es mayor. Esto es debido a que, cuando los dos objetos están cerca, el asistente hace que el robot recorra las distancias de aproximación en un número menor de movimientos que en el planificador original, recorriendo una mayor distancia por cada movimiento. Cuando los objetos se alejan, las zonas de

movilidad se hacen más pequeñas obligando al robot a moverse un poco más suave, lo que incrementa el tiempo de la planificación.

El número de nodos y de movimientos en la solución, que arroja la planificación, es menor por el hecho de que cuando en la ruta hay una gran distancia que recorrer, sin obstáculos cerca, el robot ejecuta menos movimientos y por lo tanto hay menos puntos en la trayectoria.

El costo de la solución no presenta este mismo síntoma. En el gráfico se observa que los costos de la solución para ambos planificadores es similar. Esto parece sugerir que el costo de la solución es independiente del grado de discretización escogido en cada punto y que es dependiente de la dirección de evasión que escoja el método. Entonces, el costo de la trayectoria difiere para los dos métodos cuando alguno de ellos escogió una dirección de evasión distinta en algún momento de la planificación.

Para la prueba de Distribución vs. Desempeño se puede concluir que las diferencias en los resultados son uniformemente variables porque en Choque y Arrastre siempre hubo un intento fallido antes de escoger la dirección final que el asistente escogió directamente, razón por la cual el planificador de Choque y Arrastre se tomó un poco más de tiempo y de movimientos, mientras intentaba por otra dirección por la cual no pudo evadir el obstáculo.

Con respecto a la prueba de Rango de movilidad vs. Desempeño, las diferencias de los resultados son más amplias porque cuando Choque y Arrastre escoge una dirección de evasión no sabe si más adelante la articulación o articulaciones involucradas en el movimiento llegarán a su límite, situación en la cual se devuelve para intentar por otra dirección. Si la distancia recorrida entre la escogencia de dicha dirección y la falla en la evasión es considerable, todos los parámetros del desempeño se elevan considerablemente.

Sintetizando, con las pruebas realizadas se puede concluir que el costo de la solución depende del orden de aplicación de las direcciones de evasión lo que indica que depende de las medidas del estado actual del mundo local. Por otro lado, el tiempo de planificación, el número de nodos y el número de movimientos dependen tanto de este parámetro como del grado de discretización del EC, es decir, que son dependientes del estado actual del entorno y del nivel de libertad de movimiento en cada posición de la trayectoria.

6. Conclusiones

Se ha propuesto y desarrollado un asistente de planificación con capacidad de automatizar la definición de los parámetros de entrada a un planificador de trayectorias libres de colisión. Este planificador, en su búsqueda de una trayectoria, realiza dos tipos de movimientos: aproximación al objeto de choque y evasión del mismo. Para cada uno de estos movimientos, el asistente apoya al planificador definiendo el parámetro necesario: discretización en el espacio de configuraciones para la aproximación y el orden de

aplicación de las direcciones de evasión para el arrastre. Después de llevar a cabo el desarrollo del asistente de planificación se llegan a ciertas conclusiones que se exponen a continuación:

- Cuando se definen los parámetros de planificación con base en información del entorno y del robot, se mejoran las medidas de respuesta del planificador: el número de nodos, el número de movimientos y el tiempo de planificación disminuyen al igual que el número de intentos fallidos. El costo de la solución se mantiene constante a no ser que cada planificador escoja una dirección de evasión distinta.
- El asistente de planificación mejora el método de Choque y Arrastre porque, con su ayuda, encuentra siempre una solución (si la hay) ya que planifica con un mínimo de discretización en los lugares de la escena donde es necesario.
- Los objetivos perseguidos por el asistente y los resultados obtenidos justifican incluir directamente, en planificadores futuros, e implementar en los actuales, un asistente de tal naturaleza.
- El asistente de planificación es un aporte al estudio de la planificación de trayectorias ya que ayuda a convertir un planificador en un elemento con un grado de inteligencia más alto.

7. Bibliografía

[1] Toro, J., Morales, J., Villalobos, J., "*Planificación de Trayectorias usando Estrategias de Evasión en el Espacio de Configuraciones*", Memo de investigación, Universidad de los Andes, 1994.

[2] Toro, J., "*Planificación de Trayectoria para Robots Manipuladores*", Tesis de Magister, Universidad de los Andes, 1994.

[3] Villalobos, J. "*MSIM: Un Ambiente de Programación de Robots con Capacidad de Planificación de Tareas*", Proc. CLEI'91.

[4] Brooks, R. "*Solving the Find-Path Problem by Good Representation of Free Space*", IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, No. 3, marzo-abril 1983.

Estudio de algunos sistemas CACSD

Rodrigo Moreno Serrano
Universidad de los Andes
e-mail: ro-moren@uniandes.edu.co
José Tiberio Hernández
Universidad de los Andes
e-mail: jhernand@uniandes.edu.co

Resumen

En este articulo se pretende mostrar el estado del arte en sistemas de apoyo al diseño de sistemas de control mostrando los aspectos que se deben tener en cuenta para el diseño de un sistema de este tipo. Al final se muestran los campos de investigación en este tema.

Palabras y frases clave

CACSD, CACE, CAD, control de procesos, diseño de sistemas de control, sistemas de apoyo al diseño. ingeniería del control asistida por computador, diseño de sistemas de control asistido por computador.

1. Introducción

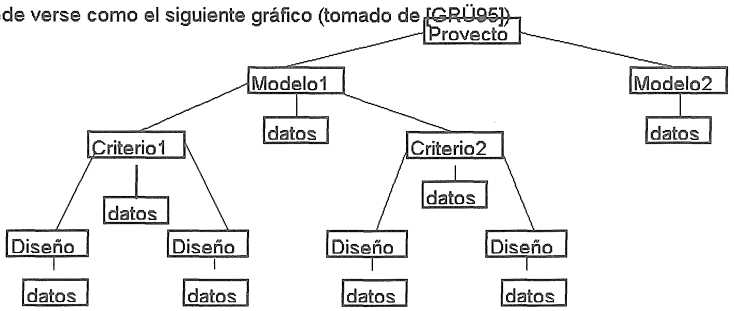
A medida que se ha desarrollado la ingeniería del control, se han desarrollado métodos cada vez más maduros que permiten diseñar mejores controladores teniendo en cuenta muchos más factores del proceso. Sin embargo, estos son extensivos en cálculos numéricos. Paralelamente, los problemas a solucionar crecieron en el número de variables (interrelacionadas) a controlar en el proceso. Por estas razones, se crean herramientas de apoyo por computador al diseño de sistemas de control.

En este documento se pretende hacer una revisión comparativa del estado del arte en aplicaciones de ayuda al diseño de sistemas de control de procesos. En un futuro se piensa estudiar la posibilidad de realizar una herramienta de este tipo usando la experiencia y el trabajo anterior del grupo de investigación DFAC de la universidad de los Andes en el diseño e implantación de sistemas CAD/CAM (e.g. [HER94]) y teniendo como referencia esta revisión.

En la siguiente sección se hará un breve recuento de las actividades del ingeniero de control cuando se enfrenta a un problema (para tenerlo en cuanta en el momento de evaluar las aplicaciones). En la tercera sección se darán los lineamientos generales que debe tener una herramienta de apoyo al diseño de sistemas de control. En la cuarta parte se comentarán los diferentes servicios que debería ofrecer al usuario este tipo de herramientas. En la quinta parte se dará una clasificación de los métodos de diseño usados en estas herramientas. En la sexta parte se hará una breve exposición de los productos estudiados. En la séptima parte se hará un análisis de los sistemas estudiados en la sexta parte y por último se darán algunas conclusiones del trabajo.

2. Proceso de diseño de sistemas de control

Cuando el diseñador se enfrenta al diseño de un sistema de control tiene que realizar básicamente tres actividades: diseñar el modelo del proceso, diseñar el sistema de control en sí y analizar todo el conjunto (o por partes) para verificar la validez de las restricciones planteadas en el diseño. En el proceso de diseño, estas tres actividades se entremezclan y no son independientes puesto que la una requiere de la otra. Una forma de ver el proceso de diseño típico puede verse como el siguiente gráfico (tomado de [GRÜ95]):



En cuanto al diseño en sí, en [NG93] se propone que el proceso es un ciclo en el que se genera un número de soluciones candidatas, se evalúa cada una de ellas y se critican los resultados de esto de lo cual pueden salir nuevas especificaciones para el siguiente ciclo o se puede decidir parar la búsqueda de soluciones. Este proceso es llevado a cabo con la colaboración tanto del diseñador como del computador.

Un sistema CACE (ingeniería de control asistida por computador) intenta ser una herramienta más genérica que una CACSD (porque la última estaría dedicada solo al diseño), sin embargo, adelante se tratarán las palabras CACE y CACSD como sinónimas porque están intimamente relacionadas.

3. Clasificación de los métodos de diseño de sistemas de control

En [JAM92] se clasifican los métodos en tres dependiendo del tipo de sistema al que aplica: métodos para sistemas SISO (single input - single output), métodos para sistemas MIMO (multiple input - multiple output) y métodos para sistemas de muy alto orden.

Por otra parte, en [NG93], los métodos de diseño son clasificados en tres grandes grupos dependiendo de la cantidad de información analítica que se posea del sistema:

- Métodos de síntesis: se pueden aplicar cuando el problema está bien formulado en el sentido de que las especificaciones no son ambiguas y el problema planteado es implementable.
- Métodos basados en procedimientos: se pueden aplicar cuando las especificaciones no son exactas pero se tiene una buena cantidad de experiencia en ese tipo de problemas (e.g. usando métodos iterativos).
- Métodos basados en búsqueda: consiste en que dado un conjunto de clases de controladores, se busca cual es el que más se ajusta a los requerimientos del problema y con cuales parámetros.

4. Requerimientos básicos de un sistema CACE

Según [Mac89] cualquier sistema de ayuda por computador (para cualquier área de la ingeniería) debe permitir al usuario las hacer las siguientes cosas:

"construir, analizar, navegar, buscar, comparar y evaluar, razonar y hacer hipótesis, sintetizar, diseñar, manipular y modificar, experimentar, catalogar, grabar y restaurar".

Mientras que el ambiente debe ser:

"fácilmente comprensible por un individuo, amplio en el dominio en que trabaja, modular (con un número manejable de partes), predecible en su comportamiento, integrado y con relaciones coherentes entre sus partes, útil con acceso eficiente a la información relevante, tolerante a errores y debe permitir ver el efecto de ellos, extensible y adaptable, y autodocumentado."

Un sistema CAD para sistemas de control debe ayudar al ingeniero a realizar tres tareas fundamentalmente ([GRÜ95]): modelaje de la planta a controlar, análisis y simulación tanto de la planta como del conjunto planta/control y diseño del control de la planta; dado que estas son las actividades que el diseñador realiza cuando se enfrenta a un problema de control.

Debido a la gran cantidad de acciones que debe permitir un paquete CACSD, algunos han decidido proponer arquitecturas abiertas en las cuales se pueden introducir un número ilimitado de herramientas que se interrelacionan entre sí usando las facilidades que se ofrecen en el ambiente donde se implantan [GRÜ95] [RUT95].

Los productos que tienen arquitectura abierta deben cumplir los siguientes requerimientos [BAR93]: debe proveer un buen conjunto de facilidades básicas de modelaje, debe proveer un solo paradigma (e.g. la aplicación debe seguir el paradigma OO), debe proveer herramientas simples fácilmente integrables, debe ofrecer extensibilidad y configurabilidad (i.e. adicionar nuevas herramientas fácilmente y debe poder configurarlas el diseñador), debe ofrecer interoperabilidad entre herramientas, debe proveer soporte para otras aplicaciones (abiertas), debe asegurar su apertura (i.e. no debería ser necesaria la inclusión de herramientas propietarias).

5. Componentes de un CACE abierto

Teniendo en cuenta que la idea de un sistema de este tipo es ofrecer interoperabilidad entre aplicaciones de tal forma que se puedan integrar fácilmente nuevas herramientas especializadas y crear nuevas con las ya existentes, en [BAR93] se propone un esquema en el cual un CACE está compuesto de los siguientes componentes:

5.1. Servicios de modelaje

En [BAR93] se propone dividir los servicios de modelaje en tres capas:

- Bus de datos con modelo neutral

En este nivel se debe manejar un modelo lo más general posible en el sentido de que pueda ser transformado a cualquier otro tipo de modelo fácilmente. Las herramientas deben comprender ese modelo puesto que se deben comunicar entre sí mediante este bus de datos. El esquema de representación más usado para este propósito es el

de ecuaciones diferenciales algebraicas porque es el que más se acerca a este propósito. Un lenguaje que permite expresar dichas ecuaciones es DS-Block (DS-Block permite expresar dichas ecuaciones usando bloques de control) [OTT92]. En este documento vamos a suponer que el esquema de representación usado como modelo neutral son las ecuaciones diferenciales algebráicas.

- Descripción del modelo en alto nivel

El poder de representación de las ecuaciones diferenciales algebráicas es muy grande y esta es la razón por la cual se usa como bus de datos con modelo neutral. Sin embargo, los lenguajes que se usan en éste nivel no son prácticos para los dieñadores de los modelos porque no facilitan la estructuración de los modelos (i.e. no tienen facilidades para dividir los modelos en submodelos). Por la misma razón, los lenguajes del bus de datos con modelo neutral no ofrecen facilidades para la reutilización de modelos.

Por estas dos razones, una herramienta CACE debería proveer un lenguaje de modelaje de más alto nivel al diseñador que le permita dar una mejor estructura a sus modelos, de modo que pueda identificar las partes de la planta en el modelo de una forma más clara. Además debe permitir ver al diseñador una parte de la planta en el modelo con el nivel de detalle que se desee.

Por otro lado, estos lenguajes deben proveer una mayor reutilización de los modelos.

Los paquetes CACSD en la actualidad ofrecen lenguajes de alto nivel que se basan en bloques que pueden contener a su vez subbloques y ,por otro lado, se encuentran los lenguajes OO como Omola.

- Objetos de datos de control (CDO)

Algunos métodos de análisis y diseño de sistemas de control usan y generan datos de tipos como ecuaciones de estado, funciones de transferencia, señales en tiempo y frecuencia (éstas dos últimas no pueden ser representadas fácilmente con las dos capas anteriores), etc, sin importarles los detalles del modelo del sistema. Si solo se tuvieran las dos capas descritas anteriormente, las herramientas que manipulan estos tipos de datos tendrían que tener traductores entre los modelos que se manejan en las otras dos capas y estos tipos de datos. Para centralizar este tipo de traducciones en un solo punto, se recomienda tener una capa de objetos de datos de control.

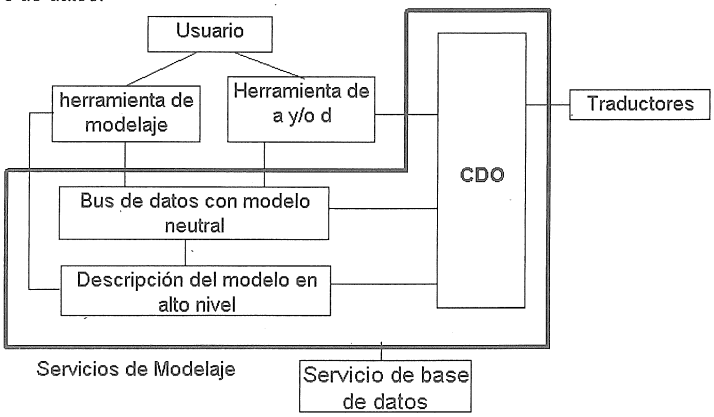
En las aplicaciones estudiadas hay dos formas de implementar esta capa: la primera es definir TADs por cada objeto y la otra forma es crear clases en el esquema de clases de la base de datos OO (si esta disponible).

Adicionalmente, al igual que en las herramientas CASE, se debe proveer un mecanismo de versionamiento de los modelos para ayudar al refinamiento del mismo por parte del diseñador.

Las tres capas se pueden ver en la siguiente figura (tomada de [BAR93]).

5.2. Servicios de manejo de base de datos

Todo el versionamiento de modelos, diseños, etc. deberían guardarse en bases de datos que puedan ser accedidas en cualquier momento por el (los) diseñador(es), por ello es recomendable apoyarse en un manejador de bases de datos.



5.3. Servicios informáticos

Existen otros servicios que debe ofrecer una herramienta CACE que dependen más de la arquitectura informática de las aplicaciones que de los servicios CAD del paquete. Sin embargo, estos deben ser tomados en cuenta al implementar una herramienta de este tipo. A continuación se describen algunas características de ellos:

5.3.1. Servicios de manejo de tareas

Debido a que se tienen diferentes herramientas en el ambiente, la aplicación debe tener un manejador de tareas cuya función es enrutar los comandos que le llegan a las herramientas a las que les corresponde ejecutarlas. De esta forma, este manejador provee la integración de las diferentes herramientas con el ambiente de la aplicación.

5.3.2. Servicios de interfaz con el usuario

En cuanto a interfaz se refiere, debe cumplir los requerimientos dados en el apartado 4. Cada herramienta puede tener su propia interfaz pero deberían ser similares para no confundir al diseñador.

5.3.3. Servicios de comunicación entre aplicaciones

Estos servicios permiten comunicar a todas las herramientas y el ambiente entre sí. Usar modelos estándares como el OSI facilita la distribución de la aplicación en una red de computadores. De esta forma el manejador de tareas se apoyaría en esta capa dando mayor independencia a las herramientas.

5.4. Herramientas

Estas herramientas deberían ser abiertas en el mismo sentido que se ha venido discutiendo para integrarlas fácilmente entre sí y con el ambiente de la aplicación. Entre los tipos de herramientas que se dispondrían estarían:

5.4.1. Herramientas de simulación

Estas herramientas, en la mayoría de casos necesitan conocer el modelo del sistema y/o el modelo del control además de las condiciones iniciales del sistema y los parámetros específicos del simulador. También pueden tenerse simuladores que calculen diagramas típicos de sistemas como el cálculo del diagrama de Nichols.

5.4.2. Herramientas de métodos específicos de diseño

Las herramientas de este tipo ayudan al ingeniero a sintetizar compensadores usando diferentes estrategias y métodos. Deberían haber herramientas de cada uno de los tipos de diseños estudiados en el apartado 3 para el diseño de controladores.

5.4.3. Herramientas de análisis numérico

Aquí están muchos de los métodos de análisis de todos los tipos de sistemas.

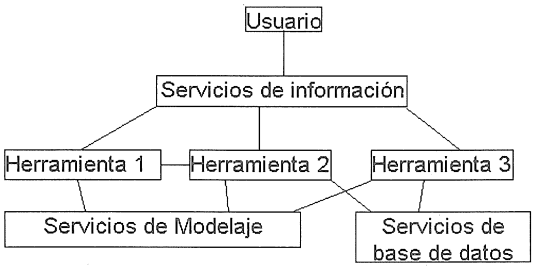
5.4.4. Herramientas de manipulación simbólica

Las herramientas de este tipo ayudan al análisis de sistemas no lineales, sistemas lineales (e.g. cálculo de matrices inversas en función de la frecuencia) y sistemas discretos y proveen soporte a sistemas expertos.

5.4.5. Herramientas de documentación

Las herramientas de este tipo ayudan al ingeniero a mantener información sobre los proyectos, además de ayudarle a generar reportes, resultados, etc.

De esta forma, la estructura de una aplicación CACE sería como la que se muestra en la figura:



6. Estudio de productos CACSD

En [JAM92] se dividen los sistemas CACSD en dos grandes bloques: los basados en MATLAB y los que no se basan en él.

6.1. Sistemas basados en MATLAB

En los sistemas basados en MATLAB como MATRIXx, CTRL-C, SCILAB y el mismo PC-MATLAB; la ayuda al modelaje de la planta es mínimo puesto que el ingeniero de control debe convertir el modelo de su planta al lenguaje matricial que entienden estos paquetes. Esto tiene la ventaja de que la herramienta maneja solo un tipo de

dato, las matrices complejas, y así desde el punto de vista de la aplicación, es más fácil dar más ayuda en los otros dos campos que debe cubrir (análisis y diseño). Sin embargo tiene las siguientes desventajas:

- al cambiar de espacio de representación (al pasar del modelo dado por el modelador al modelo equivalente matricial) se puede perder la semántica del modelo, además de que dependiendo del modelo original, la transformación al modelo matricial puede no ser exacto y así perder información del proceso.
- el ingeniero debe darle una interpretación a los datos del paquete ajustados a su problema (e.g. los vectores en ciertos contextos pueden referirse a polinomios y en otros casos si se refieren a vectores).
- el mantenimiento del modelo puede ser engorroso.
- es muy difícil reutilizar el modelo en otros problemas similares.
- la definición de bloques no lineales no es muy sencillo.
- no es fácil manejar las técnicas de diseño que se basan en la representación del sistema en el dominio de la frecuencia.

Hay algunas herramientas basadas en MATLAB que lo usan como motor de procesamiento numérico pero que permiten hacer modelaje por diagramas de bloques de forma amigable al usuario y algunos tienen una biblioteca de bloques genéricos en los cuales sólo se ajustan unos cuantos parámetros para definirlo (e.g. SIMULINK).

6.2. Sistemas no basados en MATLAB

Los demás CACSD tienen su propio lenguaje de especificación del modelo, como es el caso de Omsim [MATT95], o son más un ambiente de integración de herramientas externas de forma que el módulo de modelaje puede ser externo, como es en cierta forma ANDECS [GRÜ95].

6.2.1. CADACS (Computer aided design and analysis of control systems) [CAD95]

CADACS es un producto desarrollado en la Universidad Ruhr Bochum de Alemania. Es un sistema abierto en el sentido en que el usuario puede diseñar sus algoritmos e incluirlos en la biblioteca de funciones del sistema. Posee el servicio de base de datos en donde se guardan los modelos de la planta y todos los datos necesarios para trabajar en el problema de control (no maneja versionamiento). Tiene un módulo de tiempo real para el cual tiene un control adaptativo y un extractor de señales que es útil para la síntesis de modelos.

De los sistemas estudiados, es el que más facilidades ofrece para el modelaje: ofrece 10 tipos de representación y tiene una amplia biblioteca de utilidades de extracción de módelos que usan aproximaciones y estimación de parámetros. También provee algunas utilidades de diseño tanto del control como de observadores.

Ofrece herramientas básicas de procesamiento como funciones de manejo y análisis de señales, funciones de transferencia, respuestas de frecuencia, variables de estado y matrices de fracciones polinomiales.

Una de sus desventajas es que el lenguaje que usa es bastante críptico y dificulta la interacción con el usuario. El lenguaje es interactivo e interpretado.

6.2.2. Omsim [MATT95][AND95]

Omsim es el simulador del lenguaje orientado por objetos de modelaje de sistemas dinámicos "Omola". Ambos fueron desarrollados en el instituto de tecnología de Lund en Suecia. Omola es orientado a bloques de control, i.e. la unidad básica de modelaje es un bloque de control.

Omola es un lenguaje orientado por objetos de tal forma que conceptos como herencia, polimorfismo, etc. se encuentran disponibles en su sintaxis.

En Omola es posible definir clases de bloques. Los atributos de estos pueden ser de varios tipos:
- objetos de otras clases (estos atributos se llaman componentes)
- variables del bloque (posiblemente para ser usadas como parámetros)
- ecuaciones que describen el comportamiento del bloque.
- conexiones entre terminales (entradas o salidas) del bloque y sus subbloques
- definición de eventos (discretos)

En Omola el comportamiento de un bloque de control no está dado solamente por las ecuaciones que lo modelan, sino que además se deben añadir una serie de restricciones (que no hacen parte del modelo) para saber cuales variables de las ecuaciones se deben calcular y cuales están dadas. Así por ejemplo, la ecuación que modela totalmente el movimiento de un cuerpo es:

$$m\, d^2x/dt^2 = F$$

sin embargo, si se quiere simular el bloque correspondiente, hay que agregar dos restricciones (usando herencia por especialización) para despejar la otra . Por ejemplo: m=mo y x=xr(t) y al simular se despejaría la fuerza. Adicionalmente se permite definir las variables de entrada y salida cuando se están creando las clases contendedoras de la clase en cuestión.

Sin embargo ellos mismos admiten que estas dos facilidades del lenguaje son difíciles de resolver para el caso general puesto que implica la reorganización de las ecuaciones para poder resolverlas (si es posible) y esto a su vez implica, para efectos prácticos, la limitación del tipo de ecuaciones permitidas en el lenguaje. En Omsim implementan estos conceptos con algunas restricciones por las razones dadas (i.e. no funciona para cualquier ecuación en la versión actual). tiene la restricción de que las ecuaciones que modelan el bloque deben ser diferenciales algebráicas.

Uno de los problemas que se presentan es la necesidad de definir valores iniciales para efectos de simulación, pero en muchos casos algunos de estos valores no están disponibles fácilmente al usuario. Omsim tiene el lenguaje OCL (Omola command language) para hacer guiones para procesamiento de una secuencia de comandos de Omsim en batch. No tiene servicio de base de datos y no es abierto en todos los servicios de la discusión hecha arriba.

6.2.3. CAMeL (Computer aided mechatronic laboratory) [RUT95]

CAMeL es un ambiente basado en procesos para sistemas de control. Es un CACSD abierto en el sentido descrito arriba. Fue desarrollado en la Universidad de Paderborn de Alemania.

Tiene un lenguaje de modelaje llamado DSL (dynamic system language) que pretende ser lo más genérico posible en el sentido de que pueda modelar prácticamente cualquier sistema. Además DSL tiene facilidades de integración. Este lenguaje es orientado a bloques de control, i.e. la unidad básica de modelaje es un bloque de control.

DSL no ofrece un lenguaje orientado por objetos sino que simplemente para cada tipo de bloque define una serie de parámetros, una serie de entradas y salidas (aquí si se diferencian), una serie de variables de estado y un conjunto de ecuaciones de estado que definen el comportamiento del bloque. Usando DSL se pueden definir bloques que estan compuestos por otros definiendo la forma de interconexión entre ellos.

Solo permite que las ecuaciones que modelan el bloque sean diferenciales y/o algebraicas (en la forma de ecuaciones de estado).

Ofrece interfaces de comunicación con otros paquetes CACSD como MATLAB y otros. Para interactuar con las herramientas externas de control, ofrece dos formas de organizar los procesos de diseño: una en la cual el usuario define un pseudocódigo para definir procesos batch y la otra forma que ofrece es ejecutar cada acción en un proceso diferente y definir la sincronización entre ellos. Esta última forma ayuda a la distribución del producto y al uso de máquinas paralelas (usa PVM para ello). Adicionalmente, posee un modulo de simulación de ecuaciones no lineales llamado TRANSIENT. No tiene servicio de base de datos.

6.2.4. ANDECS (Analysis and design of controlled systems) [GRÜ95]

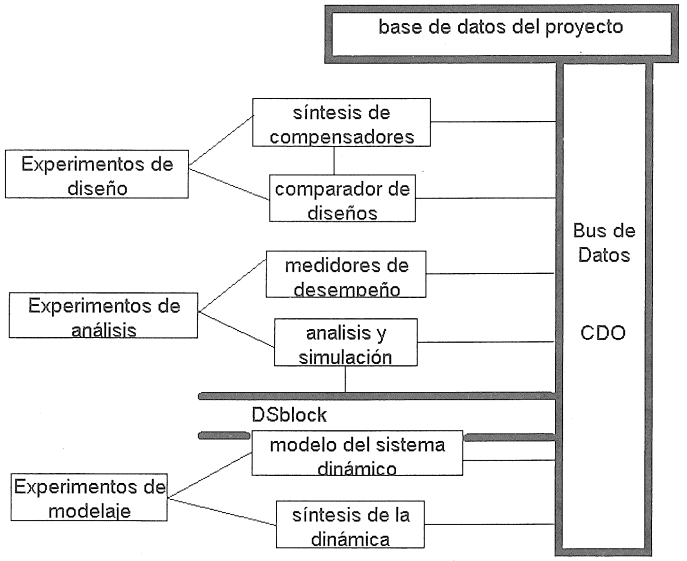
ANDECS es un ambiente basado en procesos para sistemas de control. Es un CACSD abierto en el sentido descrito en las primeras secciones. Fue desarrollado en el instituto de robótica y sistemas dinámicos DLR de Alemania.

Posee una base de datos llamada RSYST que permite la persistencia de datos complejos. También maneja un sistema muy básico de versionamiento. La comunicación entre las herramientas se hace por medio de la base de datos únicamente. Se usa PVM para la comunicación entre procesos.

Para definir modelos la idea es que el usuario lo diseñe en el paquete de su predilección y luego lo convierta a objetos de datos de control (CDOs). Estos CDOs son los que van en la base de datos. A otro nivel se tiene DSblock (Dynamic system block) [OTT92] que es un protocolo estándar para comunicar los métodos numéricos genéricos con la definición de los modelos. En este punto es que se encuentran los conversores a otros dominios de modelaje porque es en cierta forma un modelo computacional neutral (de hecho se tiene un traductor a Dymola, el antecesor de Omola).

Posee dos vías de acceso a la biblioteca de funciones numéricas básicas: una es la biblioteca RASP y por otra parte tiene un interpretador de MATLAB llamado Amatlab (ANDECS_MATLAB) por donde se puede tener acceso a ellas. Además, Tiene un módulo llamado MOPS (multi objective programming system) que permite al usuario crear archivos de procesamiento en batch (con posible interacción con el usuario) para hacer todo el proceso de diseño de los compensadores siguiendo el método que se describió en la segunda sección.

La estructura global del producto se presenta en la siguiente figura.



7. Análisis de los productos

7.1. Lenguajes de modelaje

Cada uno de los lenguajes estudiados tiene ventajas sobre el otro. Por ejemplo la mayor ventaja de Omola es su orientación por objetos que permite la reutilización de los modelos más fácilmente.

El problema de Omola es que al pretender dar la mayor cantidad de genericidad al modelo (para aprovechar al máximo las ventajas de un lenguaje OO) se complica un poco la comprensión del modelo, por ello sería probablemente más útil exigir más restricciones en la definición de las ecuaciones del bloque y definir totalmente las entradas y las salidas (como lo hace DSL).

DSL es un lenguaje un poco más natural (para el ingeniero de control) por cuanto es más orientado a bloques de control (que es lo que se maneja con más frecuencia en sistemas de control). Sin embargo, los modelos dados en

DSL son un poco más difíciles de reutilizar que los de Omola porque estos últimos se manejan como objetos ofreciendo mayores facilidades en la reutilización.

En cuanto a las ecuaciones de Omola, sería mejor tener la capacidad de definir métodos (quizás un método por salida) que describan el comportamiento del bloque. DSL es relativamente pobre en cuanto a las ecuaciones que permite (solo modelaje con variables de estado) y el problema es que dado que esto va en la definición del bloque, sería necesario extender el lenguaje para ello con el inconveniente de que las no linealidades no tienen una forma común de definición. Por ello el comportamiento debería estar fuera de la definición del bloques organizado por funciones que calculen cada una de las salidas del bloque y así no sería necesario diferenciar los bloques que definen bloques básicos de los que definen la interconexión de un conjunto de subbloques.

Por otra parte, en los lenguajes de modelaje debería poderse distinguir los bloques de control de los bloques de la planta y poder separarlos de forma que el cambio de estrategia de control sea mas fácil y así poder facilitar la labor de diseño asistido. A su vez, se debería poder definir esquemas de control jerárquico y descentralizado para facilitar el diseño.

7.2. Productos

Tanto CAMeL como ANDECS tienen el mismo enfoque en el sentido de crear un ambiente donde se integren herramientas externas ofreciendo servicios básicos para ello. Al estudiarlos, se puede decir que ANDECS tiene las ideas más maduradas (e.g. tiene manejo de bases de datos) aunque en algunas características, CAMeL ha profundizado la investigación (e.g. CAMeL ha hecho mayores avances en la programación de tareas puesto que permite la programación en paralelo de tareas).

Por el contrario, Omsim pretende ser una herramienta de modelaje y simulación que podría incluirse en los dos primeros y es abierto en el sentido de ofrecer modelos genéricos que podrían integrarse con otros paquetes (como ocurre en ANDECS) hechos en Omola.

Por otra parte, CADACS pretende ofrecer la mayor funcionalidad posible dentro de él mismo, permitiendo además la programación de nuevos algoritmos por parte del usuario. En este sentido es menos abierto que los dos primeros, pero el hecho de tener un buen número de algoritmos y servicios lo hace, en cierta medida, más fácil de manejar. Adicionalmente tiene el módulo de tiempo real que puede ser muy útil tanto para la captura de señales como para la implementación de ciertos controladores, además de tener la posibilidad de crecer a sistemas supervisores de procesos.

A continuación se muestra una tabla comparativa de los productos estudiados bajo los parámetros estudiados en las primeras secciones del documento. También se muestra que el único paquete con módulo de tiempo real es CADACS que puede llegar a ser interesante en la implementación de controles por computador.

Criterio	Basados en Matlab	CADACS	Omsim	CAMeL	ANDECS
1. Servicios de Modelaje					
Modelo neutral	si para sistemas lineales	si	si	si	si
Lenguaje para modelo neutral	Matlab	no	no	no	si
Lenguaje de alto nivel	la mayoría ofrecen agrupamiento de bloques	no	si	si	externo
CDOs	si usando convenciones sobre las variables de Matlab	no	si	si	si
2. Servicios de manejo de bases de datos	no	sin versionamiento	no	no	si
3. Servicios informáticos					
Manejo de tareas	usando Matlab	no	si	si	si
Interfaz con el usuario	si	si	si	si	si
comunicacion entre aplicaciones	no	baja	no	si	si
4. Herramientas					
Simulación	si	si	si	si	si
Análisis numérico	si	si	no	externas	externas
manipulación simbólica	no	no	no	externas	externas
métodos específicos de diseño e identificación.	si	si	no	externas	externas
documentación	no	si	no	externas	externas
5. Facilidades de tiempo real					
Monitoreo en tiempo real	no	si	no	no	no
Control adaptivo en tiempo real	no	si	no	no	no

8. Conclusiones

Como conclusiones podemos decir que un sistema de apoyo al diseño de sistemas de control debería tener los servicios que se describieron en los apartados 4 y 5 para apoyar el proceso de diseño descrito en el apratado 2. Debería ofrecer métodos de diseño para todos los tipos de sistemas (SISO, MIMO y alto orden) de todas las clases de métodos (de síntesis, basados en procedimientos y basados en búsqueda). Como se pudo ver, en la actualidad,

con muy pocas excepciones se tiene una herramienta con todas esas características y por el contrario hay herramientas que concentran su capacidad en un determinado tipo de servicios.

El corazón de la aplicación es el manejador de servicios de modelaje, por ello, éste podría ser el primer problema que se podría resolver al diseñar un sistema CACSD. Otro campo por el que se podría comenzar es en la capa de servicios informáticos para crear el ambiente de trabajo de la herramienta CACSD, aunque algunas partes de ella requiere tener conocimiento sobre la interacción requerida entre herramientas y la comunicación con los servicios de modelaje. También se pueden crear herramientas abiertas que se puedan integrar fácilmente en el ambiente CACSD. Para ello, se puede explorar la posibilidad de montar todos estos cálculos sobre plataformas tipo Matlab (que manipulan las matrices de una forma muy eficiente), por ejemplo usando el paquete Scilab.

Por último, se puede pensar en como integrar los módulos de tiempo real (e.g. implementación de controles y control adaptivo) al modelo presentado (que básicamente es el que se muestra en [BAR93]) y crear el esquema de comunicación del computador con la planta.

9. Referencias

[AND95] Andersson, M. "Omsim and Omola tutorial and user's manual". Instituto de tecnología de Lund, Marzo 1995(disponible via ftp anónimo).

[BAR93] Barker, A. et al. "Open arqchitecture for computer-aided control engineering". IEEE control systems, Abril 1993.

[CAD95] "CADACS - a system for computer-aided design and analysis of control systems". Univesidad Ruhr Bochum, 1995 (disponible via ftp anónimo).

[GRÜ95] Grübel, G. "The ANDECS CACE framework". IEEE control systems, Abril 1995

[HER94] Hernández, J. Morales, J. "PICAS: un CAD de propósito específico basado en rasgos para diseño de piezas quasi-axisimétricas". Revista Sistemas de ACIS, Enero-Marzo 1994.

[JAM92] Jamshidi, M. Mahmoud, T. Bahram, S. "Computer Aided analysis and design of linear control systems". Prentice Hall. 1992.

[Mac89] MacFarlane, A. Grübel, G. "Future design environments for control engineering". Automatica, Vol 25. 1989.

[MAT95] Mattsson, S. Andersson, M. "Ideas behind Omola". Instituto de tecnología de Lund, 1995 (disponible via ftp anónimo).

[NG93] Ng, W. "Perspectives on search-based computer-aided control systems design". IEEE control systems, Abril 1993.

[OTT92] Otter, M. "DSblock: A neutral description of dynamic systems". Open CACSD electronic newsletter, Mayo 1992 (disponible via ftp anónimo).

[RUT95] Rutz, R. Richert, J. "CAMeL: an open CACSD environment". IEEE control systems, Abril 1995

Firewall Monitoring

Jianyu You
NEC Systems Laboratory, Irving, Texas, USA
Ernst L. Leiss
Department of Computer Science, University of Houston, Houston, Texas, USA

Abstract

Securing resources against unauthorized access and/or use is a major concern of every organization that uses computer networks. To protect internal networks from external attacks, firewalls are utilized since they restrict network access while letting legitimate users have unencumbered access. Firewalls are also used to log security auditing information about connections and operations. We describe a monitor database gateway (MDBG) designed to replace older forms of firewall logging by a database system. SQL commands can be used to retrieve logged information instead of ad hoc scripts. The db application and allows secure access from other components of a firewall through the Kerberos authentication as well as other authentication methods. If the underlying database changes, only a small portion of the MDBG must be modified; the code for the other components of the firewall remains unaffected.

1. Introduction

The recent explosive expansion of computer networks has increased the dependence of organizations and individuals on the information stored and communicated using network systems, leading to a heightened awareness of the need to protect data and resources from unauthorized access and use, to guarantee the authenticity of data and messages, and to protect systems from network-based attacks. The types of attacks on the security of a computer system or network can be characterized by viewing the function of the computer system as providing information. Information flows from a source (a file, a region of main memory) to a destination (another file or user). We distinguish the following four attacks [15].

- Interruption: An asset of the system is destroyed or becomes unavailable or unusable.
- Interception: An unauthorized party gains access to an asset.
- Modification: An unauthorized party tampers with, an asset.
- Fabrication: An authorized party inserts counterfeit objects into the system.

A useful categorization of attacks is in terms of passive and active attacks:

- Passive Attacks: Eavesdropping on, or monitoring of, transmissions, to obtain information being transmitted.
- Active attacks: Modification of the data stream or the creation of a false stream.

Other attack taxonomies are discussed in [1].

To protect its systems from attacks, an organization must have security mechanisms and security services. Underlying most of the existing security mechanisms is cryptography. Very important in achieving these goals are firewalls, software/hardware systems that protect a computer network by enforcing an access control policy between two networks (see [2,13,14,17,20,21,22]). Usually, the network is protected against threats coming from an external network that cannot be trusted and from which security intrusions can originate. Protecting the network involves preventing unauthorized users from having access to sensitive data, while allowing legitimate users to have unencumbered access to the network resources.

A necessary part of a firewall is a bastion host. This is any firewall host exposed to an external network that forwards data between the internal and external networks. It is the central host in an organization's network security and must be closely monitored by the network administrators. Also important in TCP/IP (Transmission Control Protocol / Internet Protocol) networks are multi-homed hosts, hosts that have multiple network interface boards. Such hosts can route traffic between the connected network segments. A dual-homed host is a special case of a multi-homed host that has two network interfaces and has the routing functions disabled (Figure 1). A dual-homed host can be used to isolate an internal network from an external network. It is an example of a bastion host (Figure 2).

Firewalls are classified into three main categories ([2]): packet filtering, circuit gateways, and application gateways. A packet filter usually comes with screening router software, to connect to the Internet. Packet filters work by dropping packets based on

their source or destination addresses (ports). The administrator can define a list of the acceptable machines and services (based on port numbers) and a stoplist of unacceptable machines or services. Because a packet filter is able to analyze headers of the Internet's protocols (TCP/IP and UDP), TCP state flags can also be used to filter packets. UDP is connectionless and so does not retain state information. One must filter UDP based on port numbers.

Packet filtering technologies used in screening routers provide an efficient and general way to control network traffic. No changes are required to client and host applications (they operate on the IP and TCP layers), and these layers are independent of application-level issues. This simplicity has disadvantages:

(1) Because of the lack of context information, certain protocols such as UDP and RPC (Remote Procedure Call) cannot be filtered effectively. Also, auditing and alarm mechanisms are often missing.

(2) There are no good administration and user interfaces.

(3) The number of rules may be limited. Also, as the number of rules increases, the extra processing needed for every packet implies a high performance penalty.

(4) There is no way for the filter to distinguish securely one user from another.

Thus packet-filtering devices such as screening routers are often augmented by other devices called application-level gateways and circuit-level gateways (Figure 3).

Rather than using a general-purpose mechanism to allow many different kinds of traffic to flow, special purpose code is used for each desired application in **application-level gateways**. With them, one need not worry about interactions among different sets of filter rules, nor about holes in thousands of hosts offering nominally secure services to the outside. Only a chosen few programs need be scrutinized. Instead of a list of rules that control which packets or sessions should be allowed through, a program accepts the connection, typically performs strong authentication on the user which often requires one-time passwords, and then often prompts the user for information on what host to connect to. This is more limiting than packet filters and circuit-level gateways since there must be a gateway program for each application (e.g. telnet, ftp, X11, etc.). However, it provides much higher security because it can perform strong user authentication to ensure that the person on the other end of the connection is really who they say they are. Additionally, it can perform other access checks on a per-user basis (what times they can connect, what hosts they can connect to, what services they can use). Moreover, application-level gateways can be used to maintain an intelligent log of all usage of the applications. A major disadvantage of application-level gateways is that a custom program must be written for each application.

Circuit gateways relay TCP/UDP connections. The caller connects to a TCP/UDP port on the gateway which connects to some destination on the other side of the gateway. During the call, the gateway's relay program copies the bytes back and forth. In most cases, the connection service needs to be told the desired destination. A protocol describes the desired destination and service, and the gateway returns error information if appropriate. If the connection is successful, the protocol ends and the real bytes start flowing. These services require modifications to the calling program or its library. These gateways are also called "proxies". A very popular example is the SOCKS package, originally written by David Koblas [7], now maintained by NEC [23]. The SOCKS daemon runs on the firewall. Circuit-level gateways are more flexible and provide a general approach for building application gateways.

What Can a Firewall Not Do?

A firewall cannot control whatever happens after a user has passed the authentication and access checks. If a disgruntled employee properly identifies himself, properly logs into a secured machine, and then deletes files on that machine, the firewall did its job in ensuring that it was a properly authorized user.

A firewall cannot control people who go around it. That same disgruntled employee could plug his phone line into a modem attached to the network, dial up and bypass all firewall checks. For effective firewall use, network perimeter security integrity must be maintained. For more about host security, see [4].

PEDECIBA INFORMATICA

A firewall cannot protect against a data-driven attack -- attacks in which something is mailed or copied to an internal host where it is then executed. Theoretically, virus detection is undecidable [9]: no general algorithm can exist to detect general viruses. In [9], virus examples, anatomy, diagnosis, prevention and cures are extensively discussed.

Reasons for Using a Monitor Database

An activity log of the connections through a firewall provides means for analyzing network security. By maintaining a log and reviewing it periodically, statistics about services under attack and other security violations can be obtained. One can observe trends and adjust security policies to tdeal with new threats. If one knows an external site had been penetrated and the hackers had collected passwords, one can warn the users of the system. If a userid is "root" or "sys" and failed to authenticate, it is probably an intent of break-in to gain control of the computer. Repeated failures during a short time interval are highly related with password-guessing attempts. Even for connection without errors, statistical anomalies in the user profile may signal possible attacks [9]. By keeping the database for a long time, it is possible to write scripts to detect carefully paced attacks over weeks or months.

Current practice logs all activities in a flat log file. This is inconvenient because the system administrators must write their own parsing code to analyze the log. A Monitor Database Gateway (MDBG) solves this problem by storing the log data into a database supporting SQL (Structured Query Language) queries. Retrieving connection data is much easier than from an unstructured log. Additionally, a monitor database can provide information related to billing, traffic analysis, and network usage.

2. The Monitor Database

The main purpose of the monitor database is to store data of network connections through a firewall. Based on client/server software, the programs use simple function calls like TCP networking, system time and DES encryption, so it can be ported easily to other Unix machines. Specifically, our approach offers the following advantages:

1. It lets other modules of the system work without changing their codes.
2. It provides secure data storage and retrieval.
3. It implements other features not provided by the underlying database like scheduling.
4. It supports more platforms in addition to that of the DBMS in use.

Consider the modules serviced by the Monitor Database Gateway (MDBG; Figure 4). All require access to the database. Without the MDBG, when the firewall is installed for a particular database, all these modules need to be rewritten in order to communicate with the underlying DBMS. The MDBG eliminates this necessity: when the firewall has to use a new DBMS, only a small fraction of code in the MDBG needs to be modified while authentication and networking code remain same.

A firewall system's own security is crucial, because the monitoring process can itself be subverted. Some intruders may be aware of the logging mechanisms. If the DBMS used for monitoring purpose has limited network security, the database gateway can add this capability.

2.1 Authentication

Authentication is the process of a computer reliably verifying the identity of someone, some process, or some computer.

Authentication functions

Message authentication can be viewed as consisting of two levels. At the lower level, there must be some function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message. The lower-level functions may be grouped into three classes ([15]):

(i) Message encryption: the encrypted text of the message serves as its authenticator.

(ii) Cryptographic checksum: A public function of the message and a secret key that produces a fixed-length values that serves as the authenticator.

(iii) Hash function: A public function that maps a message of any length into a fixed-length hash value, which serves as the authenticator.

Message encryption can be done using conventional or public-key encryption schemes. A widely used conventional or symmetric cryptosystems is the Data Encryption Standard (DES). Complementing one bit of the key or of the plaintext will flip approximately 50% of the output bits. Thus, almost no information about the input is leaked; small perturbations in the plaintext will produce massive variations in the ciphertext. A widely used cryptographic checksum, referred to as the Data Authentication Algorithm, is based on the DES [15].

Public key cryptography is also referred to as asymmetric cryptography. It was first proposed in 1975 by Diffie and Hellman ([15]). In conventional cryptography, it is impossible to communicate with two parties without prior contact. Additionally, the number of keys needed for a complete communication mesh is very large, $O(n^2)$ keys for an n-party network. Unlike in conventional cryptography, keys are not shared. Instead, each individual generates two keys: a private key that need not be revealed to anyone, and a public key that is preferably known to the entire world.

The best known public key system is RS an ([8,15]. If A wishes to send a message to B, it encrypts the message using B's public key. When B receives the message, it decrypts it using B's private key. No other recipient can decrypt the message because only B knows B's private key. If A wants to authenticate B, A can send to B in cleartext a random number, e.g., the current time of day as known to A. B encrypts the received time of day using its own private key and sends this to A. Now A decrypts B's message using B's public key and compares it with the time when A received the message. If the delay is short, A authenticates B's identity because no one else can encrypt the time of day such that when the ciphertext is decrypted with B's public key, it yields to the cleartext time of day. If A wants to sign a message and send it to B, A applies the decryption function to the cleartext message with its private key and encrypts the result with B's public key to produce a ciphertext. B can decrypt the ciphertext using its private key, then encrypt the result using A's public key to reproduce the original message. By saving the ciphertext, B can actually supply a proof that A signed the message.

Another way to authenticate a message is using a secret key to generate a small fixed-size block of data, a cryptographic checksu, or message authentication code (MAC), that is appended to the message. This technique assumes the two parties share common secret keys. In the basic scheme, the sender computes the cryptographic checksum and the message plus checksum are transmitted to the receiver. The recipient performs the same calculation on the received message, using the same secret key, to generate a new cryptographic checksum which he compares to the received checksum. Assume that only the sender and receiver know the secret key; if the received checksum matches the computed checksum, then:

1. The receiver is assured that the message has not been altered. Because the attacker is assumed not to know the secret key, he cannot alter the checksum to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender.
3. If the message includes a sequence number (such as implemented in the MDBG), the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

Notice that a cryptographic checksum provides authentication but not confidentiality. In many contexts, messages are not secret but their integrity must be assured. In such contexts it is perfectly appropriate to transmit unencrypted data, plus an MAC.

It is often impractical to apply encryption to an entire message. Instead, we may use a secure hash function. A secure hash function must have two properties. First, its output is short, on the order of 128 bits. Second, and more important, it must be infeasible to create an input that will produce an arbitrary output value. Thus, an attacker cannot create a fraudulent message that is authenticated by means of an intercepted genuine hash value. Well-known hash functions are MD2, MD4, MD5 and SHA ([6,15]). A common way to use the hash function for authentication is to let the sender append to the message a secret key known

to the receiver and to compute the hash value of the whole string. The receiver does the same with the message and the secret key. If the computed hash value matches the received one, the receiver authenticates the sender because nobody else can produce the hash value without knowing the password.

Authentication protocols

Password-based authentication refers to a secret quantity (the password) that a user states to prove he knows it. So, the user (or a host) sends his name and password in cleartext through the communication channel. The problem with password-based authentication is eavesdropping. Many protocols were designed for an environment where eavesdropping was not a concern (rightly or wrongly), and attackers were (again rightly or wrongly) not expected to be very sophisticated. It is also used when a cryptographic algorithm is not available or expensive to perform.

Address-based authentication does not rely on sending passwords over the network, but rather assumes that the identity of the source can be inferred based on the network address from which packets arrive. This technique is safe from eavesdropping, but is subject to two other threats: a) An attacker can gain access to a node, which is authorized to use other resources. b) An attacker can impersonate the network address of trusted nodes. Depending on the environment, therefore, address-based authentication may be more or less secure than sending passwords in the clear.

Cryptographic authentication protocols can be much more secure than either password or address-based authentication. The basic idea is that a user proves his identify by performing a cryptographic operation on a quantity known by the server. The cryptographic operation performed by the user is based on his secret password. The password can be used to acquired a cryptographic key either directly, as in doing a hash of the password, or using the password to decrypt a higher-quality key, such as an RSA private key, that is stored in some place like a directory service.

Trusted Intermediaries is another efficient protocol when many entities are involved in the authentication. Assume that network security is based on secret key schemes. If the network has n nodes, each computer must know n-1 keys, one for each other system on the network. Adding a new node requires n keys to be generated and securely distributed. This is clearly unworkable in large networks and is avoided by using a trusted node known as a Key Distribution Center (KDC). If a new node is installed in the network, only that new node and the KDC need to be configured with a key for that node. If a computer wants to talk to another computer, KDC can help them authenticate each other. Kerberos is a well-known trusted intermediary which will be discussed in more detail below.

Normal authentication in the MDBG

In our MDBG, users are divided into two groups: monitor users and GUI users. Monitor users have the privileges of inserting and updating the connection data, so a strict authentication is enforced. GUI users have fewer privileges than monitor users; e. g., they cannot insert or update the connection data in the files. So their authentication can be weaker than that of the monitor users.

Password-based authentication is used when a GUI user identifies itself to the MDBG. This is adequate, because GUI users do not alter the database, and the database is accessed from the internal network (internal users are trusted). For monitor users, encryption based authentication is utilized. Each monitor user sets up a password on both server and client machines. The monitor program encrypts the timestamp and its IP address and sends them to the server which decrypts the information and checks whether the timestamp has expired and whether the IP address coincides with the source of the message.

Address-based authentication is performed for both monitor users and GUI users, to ensure that accesses to the MDBG come from the internal network. After the encrypted data are analyzed, the server checks if the IP address is in the database host file. If this succeeds, the server sends a packet for recognition. Hereafter, the server receives SQL commands from the client and returns answers.

The technique used in the MDBG to authenticate individual messages is based on a cryptographic checksum. After the output of the encrypted checksum is computed, 32 bits are taken along with 4 bytes of sequence numbers to form a block. Then these 64 bits are encoded as the MAC (Message Authentication Code) and sent to the server. On the sender end, the checksum is compared with that computed by the server and the sequence number is verified. By using this method, there is no realistic way that an attacker can replay the message since it is very difficult to alter the message. The chance that an attacker uses another message with identical checksum is only 2^{-32}.

2.2 Kerberos Authentication

Kerberos [16] has become the de facto standard for authentication in networking. It serves two purposes: authentication and key distribution. Each user and each service share a secret key with the Kerberos key distribution center. Kerberos is a trusted third-party authentication service using a protocol based on that proposed by Needham and Schroeder. Like traditional authentication methods, Kerberos relies on secret passwords to ensure the identity of a person or a machine on the network. Unlike traditional methods, Kerberos has several features for safeguarding the secrecy of the passwords. It provides a protocol which facilitates mutual authentication and which never sends passwords over the network in clear text. Also, it stores passwords only in secure Key Distribution Centers (KDC). Kerberos allows the use of encryption to secure the integrity of data sent over the network and to prevent eavesdropping.

Any user, application, or host system known to Kerberos is a principal. To use Kerberos, a user must first run the program kinit, which identifies the user to the Authentication Server (AS) that runs on the KDC. The AS returns a ticket granting ticket (TGT); it allows the user to obtain tickets to communicate with other kerberized services on the network. The TGT is encrypted using a secret key that is derived from the user's password. kinit prompts the user for the password which it transforms into the user's secret key and uses it to decrypt the contents of the TGT, including a session key which must be used for subsequent communication with the AS. The TGT and the session key are saved in the user's credential cache. Note that the user's password is never sent over the network, neither encrypted nor in cleartext.

To request a service under Kerberos, a ticket for that specific service must be obtained (Figure 5). This is done automatically by a kerberized client program for that service. The client program retrieves the TGT from the user's credential cache and sends it along with the specific service request to the Kerberos Ticket Granting Service (TGS) running on the KDC. The TGS sends the ticket for the requested service to the client program, which stores the service ticket in the user's credential cache and also presents it to the application server for authentication. Part of the ticket is encrypted using a secret key shared only by that application server and the KDC. If the server successfully decrypts that portion of the ticket, it trusts that the ticket was issued by Kerberos; the user is authenticated and the requested service will be performed.

Authentication across a realm (a collection of principals under a common Kerberos administration) boundary is also possible. An ordinary ticket will not suffice, since the local KDC will not have a secret key for each and every remote server. Instead, an inter-realm authentication mechanism is used. The local KDC must share a secret key with the remote server's KDC; this key is used to sign the local request, thus attesting to the remote KDC that the local one believes the authentication information. The remote KDC uses this information to construct a ticket for use on one of its servers.

Advantages of Kerberos

Hard to break: Since both the ticket and authenticator contain the network address of the client, another workstation can't use stolen copies of these without changing their system to impersonate the owner's network address. Also, an authenticator has a short lifetime and is only valid once.

Password protection: Passwords are never sent across the network in cleartext. They are always encrypted. Additionally, passwords are never stored on a workstation or server in cleartext.

Transparent: Ideally, the user is not aware that authentication is taking place, beyond the requirement to enter a password. The user only has to identify himself once, at the beginning of a workstation session.

Low cost: It available from MIT effectively for free and includes publicly usable encryption algorithms.
Reliable: The code is fairly mature. It is widely used and tested.

Drawbacks of Kerberos

Although Kerberos is better than the address-based authentication methods used in earlier protocols, it does have some weakness and limitations [2,17].

Most computer systems do not have a secure area where to save keys. Because a plaintext key must be stored in the initial dialog to obtain a ticket-granting ticket, a secure place is needed to store it. If this plaintext key is compromised, the entire realm can be easily compromised.

The Kerberos system was developed at MIT as part of Project Athena. The disadvantage is that the environment at MIT is unlike any other organization. Kerberos was designed to authenticate the end user - the human being sitting at the keyboard - to some number of servers. It is not a peer-to-peer system, nor was it meant for one computer system dàemon to contact another computer.

Another problem is how Kerberos handles keys on multiuser computers. Cached keys can be obtained by other users logged in to the system. In a single-user work-station environment, only the current user has access to system resources, so there is little or no need to enable remote access to the workstation. However, if the workstation supports multiple users, it is possible for others to obtained the keys.

The KDC is a single point of failure. If it fails, nobody can use anything on the network. It is possible to have multiple KDCs which share the same database of keys, but this adds complexity and cost (extra machines and replication protocols) and vulnerability (more targets to be protected).

2.3 Use of Kerberos in the MDBG

The MDBG can authenticate a user or the monitor program running on other machines with Kerberos. During the configuration of the MDBG, if Kerberos is detected on the same machine, the MDBG will be compiled and linked with the Kerberos library [11]. A database API is implemented for the monitor program. It consists of three functions. When the monitor program runs and uses the monitor database API, db_open() is called before any other access. The encoded password saved in a password file is retrieved and several Kerberos functions are called to initialize the credential cache and build the principal data structure. Immediately before Kerberos obtains TGT, the password is decoded. This ensures that if some function call fails and the program exits, no cleartext password remains in the memory. Immediately after authentication call, the password is removed from the memory, no matter whether the TGT is successfully obtained. The usual lifetime of a valid TGT of 8 hours is reduced to 5 minutes.

When an MDBG user runs the interactive SQL client (isql) and uses Kerberos to identify himself to the DBMS server, the credential cache should have a valid ticket granting ticket. Otherwise kinit should be run in order to get a TGT. After obtaining the TGT, a ticket for the MDBG service must be obtained. This is done automatically by client programs that construct communicating principals, retrieve the TGT from the default user's credential cache, and send it along with the service request to the Kerberos Ticket Granting Service (TGS). The TGS sends back the ticket for the requested service to the client programs. The client programs store the service ticket in the user's credential cache and also present it to the application server for authentication. Part of the ticket is encrypted using a secret key shared only by that application server and the KDC. If the MDBG server can successfully decrypt that portion of the ticket, it can trust that the ticket was issued by the Kerberos. The server must always authenticate itself so that the clients can be sure that it is talking to the intended server and not an impostor.

Access Control

Authentication establishes who the user is. Authorization establishes what the user is allowed to do. Sometimes the user can prove authorization without divulging his identity. In most situations, especially those involving resource access across a network, security does involve the two steps of authentication (establishing who is) and determining his authorization.

There are two access control mechanisms: discretionary access control and mandatory access control [1,3]. A discretionary access control mechanism comprises those procedures that enforce the specified mediation at the discretion of individual users, a mandatory one comprises those procedures that enforce the specified mediation, not at the discretion of individual users, but rather at the discretion of a centralized administration facility. MDBG is based on Unix, so only discretionary access control is used.

An ACL (Access Control List) for each resource could be used for authorization; however, maintaining an ACL on every resource can quickly become prohibitively expensive. A common solution is to introduce the concept of groups. However, there is still some inefficiency as each system manager is maintaining his own ACL, duplicating effort. In our MDBG, access control is performed in two steps. The first step is through the list of authorized users maintained by the MDBG and the second step is through the database management system itself by maintaining database access privileges with GRANT/REVOKE commands in SQL. There are two possibilities:

1. If Kerberos authentication is used: After the MDBG authenticates a user, it goes to the password file to retrieve the encrypted password associated with the user. This password search serves a twofold purpose: (i) if the user is not in the password file, he is not an authorized user; the authorization fails; (ii) the encrypted password is decrypted and used to log into the DBMS.
2. If normal authentication is used: Authorized users are stored in the password file; to authenticate a user, the MDBG retrieves his password in the password file. If it is not found, the authorization fails. For an authorized user, the MDBG uses his name and password to log into the DBMS.

The second step of authorization is performed by the DBMS, with finer granularity. The DBMS has a rich set of grant/revoke commands to setup a particular privilege (connect, select, create, drop, etc.) on a particular object (table, view, column, etc.). When a monitor program creates the tables by API to store monthly connections, it grants only the "select" privilege to Public while retaining all privileges for itself.

The concept of groups is not used in the MDBG, since many DBMSs do not support this concept (Sybase does, Oracle and JLT do not). Instead, users are divided into two categories: monitor users and GUI users. Monitor users are those who own the databases and can grant or revoke other user's privileges. They can also set their process's priority. GUI users can only read the database and access other resources if a monitor user grants them some additional privileges.

Priority setting

The database gateway allows user to define a priority scheduling. In fact, the database can provide real-time capabilities. Our priority setting is a rather simple illustration of this capability. Consider the following situation: The system administrator receives an alarm from the firewall and suspects an intrusion is underway. He wants to know past history of the user or host. At this moment, he issues a query to the database. Evidently, it is desirable that the database answers as quickly as possible in order to let the administrator make a correct decision. However, in this moment, some GUI user may issue a long query; thus, GUI user and system administrator will share the processor, and the answer to the administrator's query is slowed down. The system administrator should be able to use a higher priority to speed up his queries. A real-time intrusion detection system using a database is described in [10].

CPU scheduling in Unix is designed for interactive use. Processes are given small CPU time slices by a priority algorithm that reduces to round-robin scheduling for CPU-bound jobs. Every process has a scheduling priority associated with it; large numbers indicate lower priority. Processes doing disk I/O or other important tasks have negative priorities and cannot be killed by signals. Ordinary user processes have positive priorities and thus are less likely to be run than are any system process, although user processes may set precedence over one another through the "nice" command. Process aging is employed to prevent starvation, even if there is no preemption of one process by another. Since the operating system has a reliable scheduling algorithm, the MDBG uses the second approach, that is, it assigns a process priority and lets the operating system do the scheduling.

When GUI users request DBMS service, they cannot assign their own priorities; the MDBG always assigns them a priority of 0. When a monitor user requests the service, he may specify a desired priority since monitor users are trusted and the MDBG is designed to let them assign their own priorities.

Password setting

The monitor database gateway allows users to connect to the database system without going through the normal system authentication. It is important to authenticate all users. One may want to let the MDBG server have root privilege to have access to the system's password file. But if it is running on a big machine with some DBMS like Oracle or Sybase, it is unlikely the system administrator will let untested software run with the root privilege. Instead the MDBG server can login on the DBMS machine requesting user's username and password. If it does not have access to the system's password file, it may have its own password file to store (hashed) passwords. For more details and discussion, see [19].

Monitor database server

Figure 6 outlines the functional description of the MDBG server. The server allows users to specify some settings, different from default values like port number. It starts up as a daemon process and sits on a well known port (currently 6548). Upon a service request, it forks a child process to handle the SQL command, then the parent process waits for another request to arrive. When a query result is obtained, the server formats it in packets and sends them to the client. On the client side, minimal processing is required to display the data.

2.4 SQL Processing

The essential part of the MDBG server is the SQL processing. Separate modules for supported DBMSs (Oracle, Sybase, Just Logic Technology) are implemented and are described below.

Oracle: Oracle is the world's largest vendor of software for managing information with about 50% of the market share for mainframe databases. The normal programming tool to write a user application (not data entry interface, in the latter case, SQLFORMS is used) is embedded SQL for different languages; PRO*C is chosen for our implementation. In embedded SQL ([12]), before executing any SQL command, we distinguish two kinds of commands, without (non-select statement) or with output (select statement). Examples are: insert a record into the database, commit a transaction, etc.

Sybase: Sybase is the second largest database vendor. It leads the industry in client/server computing. The programming tools for user applications are Open Client Library function calls or embedded SQL. Programming with embedded SQL is similar to that of Oracle and JLT (described later). The Open Client Library is more widely available for users who use Sybase because Sybase's embedded SQL translates SQL to Open Client Library calls. Here, the MDBG is implemented with the Open Client Library ([18]). Open Client makes little distinction between select statements and nonselect statements.

Just Logic Technology: Just Logic Technology DBMS [5] is a small relational database management system. The system is cheap (about $300 for stand-alone BSDI, about $400 for client/server BSDI) and can be installed on a BSDI, SCO or Linux based PC. JLT gives users the choice of three programming interfaces to suit their own needs [5]. Because of portability, embedded SQL is chosen for the MDBG.

2.5 Monitor Database Clients and APIs

An SQL client allows a user to type in SQL commands on a remote machine and get result displayed on his screen. It allows users to specify some settings, different from the default values like hostname, port number, priority and program trace, to identify themselves to the server, to perform simple command parse and to send commands to the server, to receive query results from the server and display them neatly, and to provide some help to the user.

3. Applications of the Monitor Database

There are two ways for a firewall to store data in a monitor database. The first is to let each proxy of the firewall send data to the MDBG. An application of this approach is illustrated in [19] with the popular proxy SOCKS. The second is to implement a program (e.g., a monitor) to collect data from the proxies and to send data to the MDBG. This application is illustrated using an

NEC firewall in [19]. One of the benefits of logging network operations is statistical analysis. A graphic user interface (GUI) for network use statistics is described in [19]. This GUI retrieves data from the MDBG and displays them graphically to the user in a variety of ways.

4. A Firewall Access Definition Language

Currently, many firewalls use a list format to define the access control list. For example, SOCKS v4 [23] defines the following format for access control:

action [*=userlist] src_addr src_mask [dst_addr dst_mask] [op dst_port] [: shell_cmd]

Thus

permit *=boss,root 1.2.3.4 255.255.255.255 11.12.13.14 255.255.255.255 eq telnet

means that the SOCKS server allows requests from a local user whose effective id is either boss or root, the source IP address must be 1.2.3.4 exactly, the destination IP address must be 11.12.13.14 exactly, and the service requested must be telnet. When fields are numerous (including source and destination's IP addresses, masks, protocols, port numbers, users, access time, authentication methods, etc.), defining the access control lists is error prone and inconvenient.

Our Firewall Access Definition Language (FADL) aims at providing sequential statements familiar to system administrators to generate the access control list. Though the grammar is designed to be simple at the beginning, the language construct allows expansion to meet future requirements. Using a language for defining a firewall's access control list permits a clear description of the access through natural IF..THEN..ELSE statements. It is easier to use; e. g., application names can be used instead of specifying protocol and port numbers. It is flexible: if an additional variable has to be taken into consideration, we can define a global variable instead of adding a field to every line in the normal ACL.

In our FADL, the configuration file of a firewall under consideration has two sections, (1) Connection permission, and (2) Alarm specification. The FADL consists of only three language constructs to define these sections: assignment statement, conditional statement and alarm specification. In [19], a description of the FADL is presented, then the grammar of the language is given. A method for checking consistency among the statements is developed and the combination with alarm specifications using the FADL language construct is discussed.

5. Conclusions and Extensions

A system to provide secure data storage for a firewall has been implemented. A difficult part has been finding a way to accommodate the system for different behaviors of the DBMSs. Thus, the semantics of SQL commands should be unified. The authentication scheme, the server program, and the network functions implemented in this thesis can be reused. Only the creation of new tables and the use of appropriate SQL commands to store data needs to be changed. Another extension would be to add more capabilities to the monitor database server, for instance, a better real-time capability or the processing of application specific commands, because if the client program needs to obtain the statistics for a time interval crossing different months, several SQL command must be issued to retrieve data corresponding to each month and to consolidate the total result.

6. References

[1] E. Amoroso, *Fundamentals of Computer Security*, Prentice Hall, 1994.
[2] W. R. Cheswick and S. Bellovin, *Firewalls and Internet Security*, Addison Wesley, 1994.
[3] K. Bhaskar, *Computer Security: Threats and Countermeasures*, NCC Blackwell, 1993
[4] S. Garfinkel and G. Spafford, *Practical Unix Security*, O'Reilly and Associates, 1991
[5] Just Logic Technology, Just Logic/SQL Database Manager, (user manual), Just Logic Technology, 1995.
[6] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: PRIVATE Communication in a PUBLIC World*, Prentice Hall, 1995.
[7] D. Koblas and M. Koblas, "SOCKS", Proc. 1992 Usenix Security Symp., 77-83, 1992.
[8] E. L. Leiss, *Principles of Data Security*, Plenum, 1982.
[9] E. L. Leiss, *Software Under Siege: Viruses and Worms*, Elsevier, 1990.

[10] T. F. Lunt and R. Jagannathan, "A Prototype Real-Time Intrusion-Detection Expert System", Proceedings of IEEE Symp. on Security and Privacy, 59-66, 1988.
[11] MIT Information Systems, Kerberos V5 Application Programming Library, MIT Information Systems, 1994.
[12] Oracle Corporation, Pro*C Supplement to the ORACLE Precompilers Guide, Oracle Corporation, 1990.
[13] M. J. Ranum, "An Internet Firewall", Proc. World Conf. Systems Management and Security, 1992. Available by ftp from decuac.dec.com:/pub/docs/firewall/firewall.ps.
[14] M. J. Ranum, "Thinking About Firewalls", Proc. Second World Conf. Systems Management and Security, 1993. (http: www.tis.com/Home/NetworkSecurity/Firewalls/ThinkingFirewalls.html.
[15] W. Stallings, *Network and Internetwork Security*, Prentice Hall, 1995.
[16] J. Steiner, C. Neuman, and J. Schiller, "Kerberos: An Authentication Service for Open Networked Systems", Proc. Winter 1988 USENIX Conference, 191-202, February 1988.
[17] K. Siyan and C. Hare, *Internet Firewalls and Network Security*, New Riders Publishing, 1995.
[18] Sybase Inc., Open Client DB-Library, Sybase Inc., 1991
[19] J. You, " Firewall Monitoring Using Databases", M. S. thesis, Dep. Computer Science, Univ.of Houston, Houston, Texas, December 1995.
[20] ftp://ftp.greatcircle.com - Firewalls mailing list archives, directory: /pub/firewalls
[21] ftp://ftp.tis.com - Internet firewall toolkit and papers, directory: /pub/firewalls
[22] ftp://research.att.com - Papers on firewalls and breakins, directory: /dist/internet_security
[23] http://www.socks.nec.com - Document for SOCKS and free software, directory: /

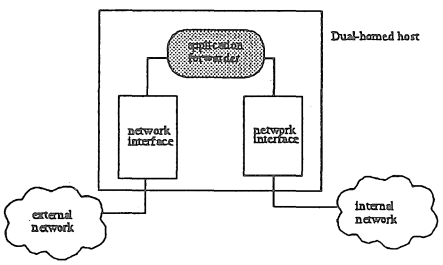


Figure 1 Dual-homed host

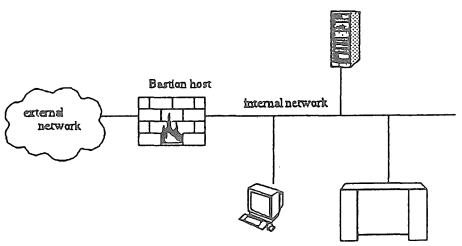


Figure 2 A Bastion host (dual-homed)

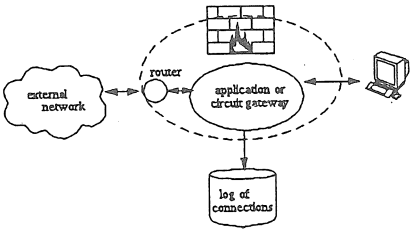


Figure 3 Application and circuit gateway

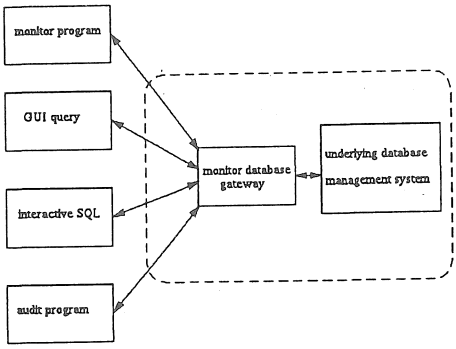


Figure 4 . Modules serviced by the MDBG

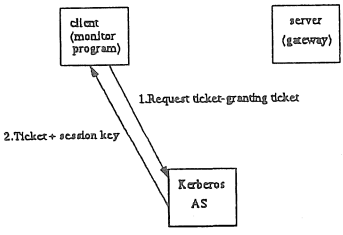


Figure 2.4 Process of getting TGT during db_open (equivalent to typing kinit)

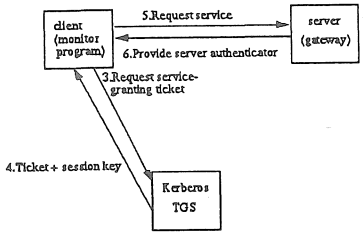


Figure 5. Obtaining a service

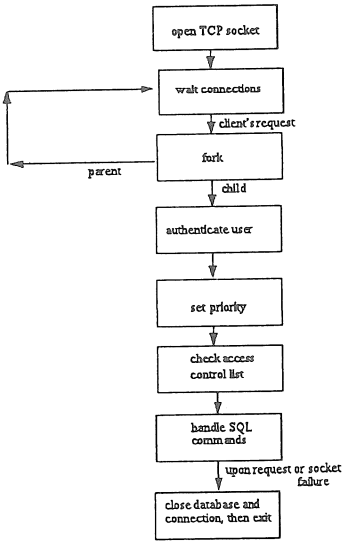


Figure 6 Steps performed by the MDBG server

A Firewall Access Definition Language

Jianyu You
NEC Systems Laboratory, Irving, Texas, USA
Ernst L. Leiss
Department of Computer Science, University of Houston, Houston, Texas, USA

Abstract

Securing resources against unauthorized access and/or use is a major concern of every organization that uses computer networks. To protect internal networks from external attacks, firewalls are utilized since they restrict network access while letting legitimate users have unencumbered access. This paper describes the design of a language to specify firewall access control lists and alarms. A procedure to check consistency among access permissions is also developed.

1. Introduction

Firewalls are hardware/software systems that allow one to isolate a subnet from a larger network in a way that restricts access to certain groups of users. Securing resources against unauthorized access and/or use is a major concern of every organization that uses computer networks. To protect internal networks from external attacks, firewalls are utilized since they restrict network access while letting legitimate users have unencumbered access. Firewalls can also be used to log security auditing information about connections and operations. For more details about firewalls, their uses, restrictions and limitations, as well as related issues, see [5].

Currently, many firewalls use a list format to define the access control list; this is quite inconvenient and can often lead to confusion. Also, when fields are numerous (including source and destination's IP addresses, masks, protocols, port numbers, users, access time, authentication methods, etc.), it is error prone and inconvenient to define the access control list. For example, SOCKS v4 [6] defines the following format for access control:

action [*=userlist] src_addr src_mask [dst_addr dst_mask] [op dst_port] [: shell_cmd]

Thus

permit *=boss,root 1.2.3.4 255.255.255.255 11.12.13.14 255.255.255.255 eq telnet

means that the SOCKS server allows requests from a local user whose effective id is either boss or root, the source IP address must be 1.2.3.4 exactly, the destination IP address must be 11.12.13.14 exactly, and the service requested must be telnet.

The Firewall Access Definition Language (FADL) defined in this paper aims at providing sequential statements familiar to system administrators to generate the access control list. Though the grammar is designed to be simple at the beginning, the language construct will allow expansion to meet future requirements.

The advantages of using a language for defining a firewall's access control list include the following:

- It offers a clearer description of the access through the use of natural IF..THEN..ELSE statements.
- It is easier to use; for example, application names can be used instead of specifying protocol and port numbers.
- It is flexible: if an additional variable has to be taken into consideration, we can define a global variable instead of adding a field to every line in the conventional access control list specification.

In our FADL, the configuration file of a firewall under consideration has two sections:

(1) Connection permission
(2) Alarm specification

The FADL consists of only three language constructs to define these sections: the assignment statement, the conditional statement, and the alarm specification.

Since the FADL is a little special purpose language, we can use criteria discussed in [1] and [2] for language design, namely, orthogonality, generality, parsimony, completeness, similarity, extensibility, and openness. These seven criteria are sketched below; they served as a guide to designing the FADL. These criteria refer to either the statements of the language or aspects of its implementation. For example, orthogonality refers to statements, and openness refers to statements and implementation.

In the context of programming languages, **orthogonality** means that different statements have different purposes, When designing a language, a single statement should be formulated for each function that the language is carry out.

The **generality** of a statement refers to the range of applications to which the statement applies. For example, a language may contain several statements to create program loops. There could be one for counting (such as the FOR-loop of BASIC), a WHILE-loop, a REPEAT-loop, and a DO...LOOP. The generality criterion would favor just one for all of the looping statements.

Another important characteristic of a programming language is **parsimony** which means that a language should not be wasteful; thus, no statement should duplicate functionality. A little language should not mimic languages with several statements to accomplish one task.

Completeness means that a language should meet all of the requirements of its specification to solve a problem.

Similarity means that the terminology used should reflect the domain of its application. The terminology and statements should be similar to those used in the problem domain of the language.

The designer and other users may have new ideas for, and enhancement suggestions to, a language in use. It is in our best interest to make it possible for the language to grow, to change, and to evolve. This is called **extensibility**.

The last criterion is that of **openness**, or the ability to share, the ability to communicate outside of one's boundaries like data transfers and external function calls.

In the next section, a description of our FADL is presented, then the grammar of the language is proposed. A method to check consistency among the statements is developed and the combination with alarm specifications using the FADL language construct is discussed.

2. Description of the FADL

Assignment statement

The variable naming is similar to that of `make`, `shell`, etc in Unix. The value of a variable is referenced by preceding its name by $. The syntax for an assignment is:

variable = value_list

The construct can be used to define lists, needed for our application. It is in a form familiar to system administrators; parsing codes from programs such as make are reusable. Two examples of the assignment statement are:

```
VAR1 = 123.445.566.323  123.344.566.666
SOURCE_LIST = .syl.dl.nec.com  .syl.sj.nec.com
              cnad.nj.nec.com \ .asl.dl.nec.com  123.455.678
```

In the future, the concatenation operation (such as in `VAR2 = $VAR1 $SOURCE_LIST`) and others may be supported. Predefined variables are used for an ACL's columns; they include:

`SOURCE`	the format is: `source address (SOURCE_MASK)`. If `SOURCE_MASK` is not specified, a default mask is computed.
`DESTINATION`	destination address. Similar to `SOURCE`.
`APPLICATION`	values can be `TELNET`, `FTP`, etc. or a triplet `(PROTOCOL, SOURCE_PORT, DESTINATION_PORT)`. Default values for services such as `TELNET`, `FTP` are defined.
`AUTH`	authentication method.
`USER`	user name.
`PERMIT`	has permission: YES or NO.
`ACCESS_TIME`	time in which access is allowed.
`ALERT_TIME`	time in which failed attempt numbers are kept.
`DENY[$SOURCE]`	number of denied requests during alert time by the source of the current attempt.
`DENY[$DESTINATION]`	number of denied requests during alert time by the destination of the current attempt.
`DENY[$USER]`	number of denied requests during alert time by the user of the current attempt.

These can be used to formulate statements.

Conditional statement

The conditional statement has the following form:

```
IF (condition)
THEN list-of-statements
ELSE list-of-statements
ENDIF
```

where list-of-statements is a sequence of assignment statements or other conditional statements (nested IF statements), and condition can be described by the following simplified grammar:

```
condition::= operand bin_operator operand
operand::= $variable | value_list
bin_operator::=  >= | <= | > | < | == | != | IN
```

The operator `IN` checks if the first operand matches some elements of the second operand (a list). Some examples are as follows:

```
IF ($SOURCE IN $SOURCE_LIST)
THEN PERMIT=YES
ELSE PERMIT=NO
ENDIF
IF ($SOURCE == 123.456.567.555)
THEN
     IF ($DESTINATION IN $VAR1)
     THEN
          APPLICATION=TELNET HTTP
          PERMIT=YES
     ENDIF
ELSE
     PERMIT=NO
ENDIF
```

Scope of variables

In order to isolate the effect of variables, the concept of block is defined; this is a set of statements enclosed by { }. Similar to C functions, variables used inside a block are local. Thus a user can define what is equivalent to a single access control line, without affecting variables in other blocks:

```
{SOURCE=123.144.145(SOURCE_MASK=255.255.255.0)
 DESTINATION=122.333.333.32
 PERMIT=YES
}
```

Variables defined outside a block are global to that block. If the authentication method is the same for the whole access control list, we can define a global authentication method and not bother to define it in every block. Variables defined as global can be used inside different blocks.

The need for checking consistency

When the number of lines in a usual access control list is small, say, 5 or 6 lines, the visual detection of inconsistency among them is easy. But as the number of lines grows above 20, visual inspection may be very difficult. Some possible errors such as those shown in the following program fragment may occur:

```
deny   source in .COM
...
allow source in .NEC.COM
```

The original goal was "deny everything from .COM but allow .NEC.COM". But here the `deny` line comes first. So `.NEC.COM` will be denied. The correct order should be

```
allow source in .NEC.COM
...
deny   source in .COM
```

The FADL compiler detects this situation using the unification concept taken from artificial intelligence. If the access control list consists of several lines, we check whether the second line is consistent with the first. Then we check that the third line is consistent with the first two lines, and so on. The worst-case time complexity is quadratic, but since it is done only once, during compiling time, it will not add computational burden later.

If we combine access control lists from two different firewalls, inconsistencies among them can be discovered by the same approach. A Host Access List can be defined based on the constructs discussed above. An example is the following:

```
AUTH=RevIp
{PERMIT=NO
 APPLICATION=HTTP
 SOURCE=.COM
 DESTINATION=101.101.101.102
}
SOURCE_LIST = 13.234.233.233 .syl.dl.nec.com .asl.dl.nec.com
{IF ($SOURCE IN $SOURCE_LIST)
 THEN
      IF ($DESTINATION == 23.222.233.233)
      THEN
           PERMIT=YES
           APLLICATION=(UDP,-,-)
      ELSE
           PERMIT=NO
      ENDIF
 ENDIF
}
```

```
{IF ($APPLICATION ==(TCP,-,23) &
    $DESTINATION == 29.444.444.555)
 THEN PERMIT=YES
 ELSE PERMIT=NO
 ENDIF
}
```

The resulting access control list is equivalent to the following in list format:

permit	protocol	auth	source	source_port	destination	dst_port
NO	TCP	RevIp	.COM	-	101.101.101.102	80
YES	UDP	RevIp	13.234.233.233	-	23.222.233.233	-
YES	UDP	RevIp	.syl.dl.nec.com	-	23.222.233.233	-
YES	UDP	RevIp	.asl.dl.nec.com	-	23.222.233.233	-
NO	-	RevIp	13.234.233.233	-	-	-
NO	-	RevIp	.syl.dl.nec.com	-	-	-
NO	-	RevIp	.asl.dl.nec.com	-	-	-
YES	TCP	RevIp	-	-	29.444.444.555	23
NO	-	RevIp	-	-	-	-

3. Grammar of the FADL

We give a formal specification of our firewall access definition language by formulating a context free the grammar .

```
acl_language --> block_list

block_list   --> assignment '\n' block_list | block '\n' block_list
                 | alarm  '\n' block_list | assignment | block | alarm |

block        --> '{' stmt_list '}'

stmt_list    --> assignment '\n' stmt_list  | conditional '\n' stmt_list
                  | assignment | conditional |

assignment   --> variable '=' atom_list

atom_list    --> atom atom_list |

conditional  --> 'IF' condition 'THEN' stmt_list 'ELSE' stmt_list 'ENDIF'
                 | 'IF' condition 'THEN' stmt_list 'ENDIF'

condition    --> condition '|' condition | condition '&' condition
                 | '~' condition | '(' condition ')' | predicate  | in_predicate

predicate    --> expression COMPARISON expression

comparison   --> '==' | '<=' |'>=' | '!=' | '<' | '>'

in_predicate --> expression 'IN' value | expression IN '(' atom_list ')'

expression   --> value | atom

alarm        --> 'WHEN' condition '{' action '}'

action_list  --> command '\n' action_list |

command      --> atom_list
```

lexical tokens:

```
atom        --> (character|digit|special)+

variable    --> character(character|digit|'_')*

value       --> $character(character|digit|'_')*

digit       --> '0' - '9'

character   --> 'a' - 'z' | 'A' - 'Z'

special     -->  '.' | ',' | '(' | ')' | '_' | ':'
```

4. Consistency Checking

One of the major purposes of the FADL is to check consistency among all the rules. To accomplish this task, predicate logic can be used (see, for example, [4]). In this section, the unification algorithm is described and a procedure to discover contradictions and redundancies in an ACL is developed. In predicate logic, we can represent real-world facts as statements written as "well-formed formulas". Consider the following set of sentences:

(1) User John was denied by host H.

(2) All users should be allowed to access H

In predicate logic, sentence (1) can be represented by

deny(John,H)

deny(John,H) captures the critical fact of the sentence but fails to capture some of the information in the English sentence, namely the notion of past tense. Whether this omission is acceptable or not depends on the intended use.

The second sentence can be represented by

allow(x,H)

where x is a variable that can take any particular value. Conventionally, variables are denoted by lower case letters, and specific values, by using upper case as the first letter.

Sometimes, it is easy to determine that two assertions cannot both be true at same time [4], for example, L and ~L. In predicate logic, this matching process is more complicated since the arguments of the predicates must be considered. For example, man(John) and ~man(John) is a contradiction, while man(John) and ~man(Spot) is not. Thus, in order to determine contradictions, we need a matching procedure that compares two assertions and discovers whether there exists a set of substitutions that makes them identical. There is a straightforward recursive procedure, called the unification algorithm, that does just this.

The base idea of unification is very simple. To attempt to unify two literals (well-formed formulas converted to Clause Form, see [4]), we first check if their initial predicate symbols are the same. If so, we can proceed. Otherwise, there is no way they can by unified, regardless of their arguments. For example, the two literals

deny(John)

sleep(John)

cannot be unified. If the predicate symbols match, then we must check the arguments, one pair at a time. If the first matches, we can continue with the second, and so on. To test each argument pair, we can simply call the unification procedure recursively. The matching rules are simple. Different constants or predicates cannot match; identical ones can. A variable can match another variable, any constant, or a predicate expression, with the restriction that the predicate expression must not contain any instances of the variable being matched.

The only complication in this procedure is that we must find a single, consistent substitution for the entire literal, not separate ones for each piece of it. To do this, we must take each substitution that we find and apply it to the remainder of the literals before we continue try to unify them. For example, suppose we want to unify the expressions

deny(x,x)

deny(y,z)

The two instances of deny match. Next we compare x and y, and decide that if we substitute y for x, they could match. We will write that substitution as

y/x

(Maybe it is easier to view this substitution as x:=y). Of course, we could have decided instead to use x/y, since they are both just dummy variable names. But now, if we simply continue and match x and z, we produce the substitution z/x. But we cannot substitute both y and z for x, so we have not produced a consistent substitution. What we need to do after finding the first substitution y/x is to make that substitution throughout the literals, giving

deny(y,y)

deny(y,z)

Now we can try to unify arguments y and z, which succeeds with the substitution z/y. The entire unification process has now succeeded with a substitution that is the composition of the two substitutions we found.

The object of the unification procedure is to discover at least one substitution that causes two literals to match. Usually, if there is one such substitution, there are many. The procedure Unify(L1,L2) returns a list representing the composition of the substitutions that were performed during the match. The empty list, NIL, indicates that a match was found without any substitutions. The value FAIL signals that the unification procedure failed.

Algorithm: Unify(L1,L2)

1. if L1 or L2 is a variable or constant, then
   (a) if L1 and L2 are identical, then return NIL.
   (b) else if L1 is a variable, then if L1 occurs in L2 then return FAIL, else return {(L2/L1)}.
   (c) else if L2 is a variable then if L2 occurs in L1 then return FAIL, else return {(L1/L2)}.
   (d) else return FAIL.
2. if the initial predicate symbols in L1 and L2 are not identical, then return FAIL.
3. if L1 and L2 have different numbers of arguments, then return FAIL.
4. set SUBST to NIL. (At termination, SUBST contains all substitutions to unify L1 and L2).
5. for i:=1 to number of arguments in L1
   (a) call Unify with i-th argument of L1 and i-th argument of L2, placing the result in S.
   (b) if S=FAIL then return FAIL.
   (c) if S is not equal to NIL then
      i. apply S to the remainder of both L1 and L2.
      ii. SUBST:=APPEND(S,SUBST).
6. return SUBST.

Note: Steps 1(b) and 1(c) are to make sure that an expression involving a given variable is not unified with that variable. Suppose we were attempting to unify the expressions

f(x,x)

f(g(x),g(x))

If we accepted g(x) as a substitution for x, then we would have to substitute it for x in remainder of the expressions. But this leads to infinite recursion since it will never be possible to eliminate x.

We now have an easy way of determining that two predicate are contradictory - they are if one of them can be unified with the negation of the other. So for example, man(x) and ~man(Spot) are contradictory, since man(x) and man(Spot) can be unified. This corresponds to the intuition that man(x) cannot be true for all x if there is known to be some x, say Spot, for which man(x) is false.

Consistency checking of an access control list

The idea developed for predicate unification can serve as a method to check consistency among the statements in an access control list. We view each line in an access control list as a predicate such as deny(...) or allow(...). Because

the ACL is checked line by line, we can defined a set of statements P which initially consists of the first line. We check whether the second line is contradictory or redundant to P; if not, we add the line to P. That is, in each iteration, we check if one line is consistent with all of the previous lines, if so, we add this line to P. Specifically, for each line, the following steps are used:

Step 1. Unify the predicate (the line) with the predicates in P, considering all its arguments as constants (not to take other value). If the unification algorithm succeeds, the line is redundant. If the unification fails, the line is not redundant.

Step 2. Unify the negation of the predicate with the predicates in P, considering all its arguments as constants (not to take other value). If the unification algorithm succeeds, the line is contradictory; proceed to the next line. If unification fails, go to Step 3.

Step 3. Unify the negation of the predicate with the predicates in P. If the unification algorithm succeeds, then the line is possibly contradictory. If the unification fails, the line is definitely not contradictory. Proceed to the next line.

The following examples illustrate the consistency checking procedure:

Example 1: The access control list consists of:

deny(x,U)
deny(H,y)

In the iteration of checking deny(H,y) with P={deny(x,U)}, Step 1 fails because U cannot be unified with y (considered a constant). Thus the second line is not redundant. In Step 2, the unification fails with the negation of deny(H,y), so it is not contradictory.

Example 2: The access control list consists of:

deny(x,z)
deny(H,y)

In the iteration of checking deny(H,y) with P={deny(x,z)}, Step 1 succeeds, because a substitution is H/x, y/z. Therefore deny(H,y) is redundant. In Step 2, the unification fails with the negation of deny(H,y), so it is not contradictory.

Example 3: The access control list consists of:

deny(x,z)
allow(H,y)

In the iteration of checking allow(H,y) with P={deny(x,z)}, Step 1 fails, so there is no redundancy. Step 2 succeeds, because there is a substitution H/x, y/z to unify the negation of allow(H,y) which is deny(H,y) with P. So allow(H,y) is contradictory.

Example 4: The access control list consists of:

deny(x,U)
allow(H,y)

In the iteration of checking allow(H,y) with P={deny(x,U)}, Step 1 fails, so there is no redundancy. Step 2 fails also, by considering y a constant. Step 3 succeeds, because there is a substitution H/x, U/y to unify the negation of

allow(H,y) which is deny(H,y) with P. Therefore allow(H,y) is possibly contradictory to deny(x,U). Intuitively, the first predicate says that we deny user U no matter on which host he is. The second predicate says that we allow every one from host H. The dilemma is, what about user U from host H?

Example 5: The access control list consists of:

allow(H,U)

deny(x,U)

In the iteration of checking deny(x,U) with P={allow(H,U)}, Step 1 fails, so there is no redundancy. Step 2 fails also, by considering x a constant (H cannot be unified with x). Step 3 succeeds: there is a substitution H/x to unify the negation of deny(x,U) which is allow(x,U) with P. So deny(H,y) is possibly contradictory. Intuitively, if the purpose of this ACL is allow user U from host H and not any other hosts, the ACL is correct. Compare this with Example 4: if Step 3 succeeds, the ACL may be correct or contradictory.

5. Alarm Specifications

Alarms are actions triggered by certain events. Our proposed syntax for the specification of alarms is:

```
WHEN (alarm_condition) { action }
```

Actions may consist of commands and shell scripts. A network administrator can invoke a command file where he defines whatever he wants.

Some examples of this construct are:

```
WHEN ($DENY[$DESTINATION] > threshold & $ALERT_TIME == time_interval)
{ mail sysadm -s "intrusion" }
```

```
WHEN ($DENY[$SOURCE] > threshold & $ALERT_TIME == time_interval)
{ add2black source=$SOURCE }
```

Note that threshold specification is a typical way for real-time intrusion detection. A real-time system for intrusion detection is found in [3]. Recall that `$DENY[$DESTINATION]` and `$DENY[$SOURCE]` are the numbers of failures to connect the current destination and the source respectively. add2black can be a shell script like

```
echo $1 $2 | cat >> blacklist
kill -HUP ratchet
```

which adds the source to the blacklist and signals to "ratchet" to re-read the blacklist. We can see that several usual attack scenarios can be detected using this construct.

Attack scenarios

Suppose that several external hosts are allowed to access a single internal host, protected by a firewall (Figure 1). Depending on a attacker's ability, he can

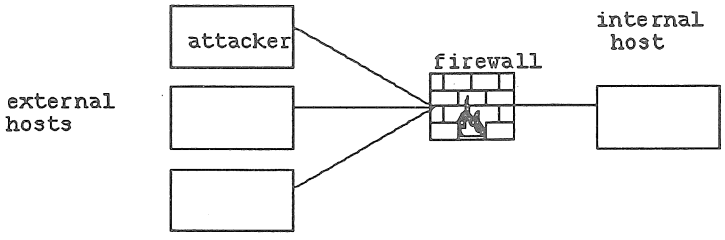


Figure 1 Attack on a single internal host

(1) attack from an external host without changing user name but with different passwords

(2) attack from an external host by changing his user name

(3) attack from different hosts at the same time

In a simplistic approach, one may define the following clauses:

```
WHEN ($DENY[$USER]>threshold & $ALERT_TIME==300)
{ add2black user=$USER }
WHEN ($DENY[$SOURCE]>threshold & $ALERT_TIME==300)
{ add2black source=$SOURCE }
WHEN ($DENY[$DESTINATION]>threshold & $ALERT_TIME==300)
{ add2black destination=$DESTINATION }
```

That is, whenever the number of failed attempts exceeds a given threshold, the user, the source, or the destination will be added to the blacklist. A better policy would be: In case (1), only the user is added to the blacklist, meanwhile other users on the same host should be allowed. In case (2), the external host where the attacker is on is added to the blacklist. In case (3), the destination should be protected by denying all the connections. (i.e., they would be added to the blacklist).

Using our construct, the following set of clauses will achieve the goal; we assume that the threshold is 5 failures during the last 300 seconds:

```
WHEN ($DENY[$USER]>5 & $ALERT_TIME==300)
{ add2black user=$USER }
WHEN ($DENY[$SOURCE]>5 & $DENY[$USER]<=5 & $ALERT_TIME==300)
{ add2black source=$SOURCE }
WHEN ($DENY[$DESTINATION]>5 & $DENY[$SOURCE]<=5 & $ALERT_TIME == 300)
{ add2black destination=$DESTINATION }
```

The first clause takes care of case (1) of the attack scenarios. In the second clause, `$DENY[$USER]<=5` means some other user has failed to connect from the same source, because the number of failures from this source exceeds 5. Therefore attack scenario (2) is assumed and the source will be added to the blacklist. Similarly, `$DENY[$SOURCE]<=5` means some other host has failed to connect to the destination, so attack scenario (3) is assumed. Now consider that we have several internal hosts as shown in Figure 2.

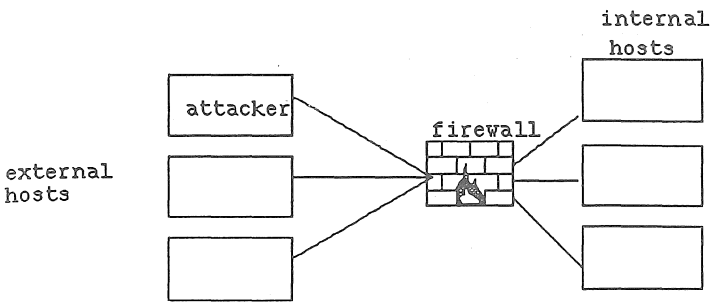


Figure 2 Attacks on several internal hosts

If the attacker tries to connect to several internal hosts without changing his user name, the first alarm clause will add him to the blacklist. If he tries to connect to several internal hosts changing his user name on a single host, the second alarm clause will add the source host to the blacklist. Finally, if he tries from different hosts changing his user name or if we have several attackers at same time, the denied connection numbers for the destinations will be high, which triggers the third alarm to add the destination to the blacklist.

6. Conclusion

We have defined a firewall access definition language to help in maintaining appropriate and easily understood and modifiable access restrictions. This provides ease of use, which in turn translates into making firewalls a more attractive approach to increasing network security. An added benefit of the approach is that consistency is checked automatically by the compiler; inconsistencies in the access definitions are often an indication of erroneous restriction formulation, which may result in reduced security.

7. References

[1] J. Bentley, "Little Languages", Communications of the ACM, vol. 29, no.8, pp. 711-721, August 1986.
[2] R. M. Kaplan, *Constructing Language Processors for Little Languages*, Wiley, 1994.
[3] T. F. Lunt and R. Jagannathan, "A Prototype Real-Time Intrusion-Detection Expert System", Proceedings of IEEE Symp. on Security and Privacy, pp. 59-66, 1988.
[4] E. Rich, E. and K. Knight, *Artificial Intelligence*, McGraw-Hill, 1991.
[5] Jianyu You and E. L. Leiss, "Firewall Monitoring", CLEI PANEL 96, Bogota, Colombia, June 3-7, 1996.
[6] http://www.socks.nec.com - Document for SOCKS and free software, directory: /

PLATAFORMA DE ANALISIS BASADA EN MODELOS COMPUTACIONALES PARA SIMULAR BISDN A NIVEL ATM

G. Isern, R.Rivas, Y. Millán

Universidad Central de Venezuela
Grupo de Investigación en Redes, Arquitectura y Sistema
Apartado 49125, Caracas,Venezuela
E-mail: isern@conicit.ve , gisern@Kuaimare.ciens.ucv.ve
Fax: 58 2 7520402

Abstract

Se presenta una plataforma de simulador de comunicaciones de BISDN a nivel ATM que sirve como soporte para estudiar sus protocolos y algoritmos para el manejo de la congestión con la finalidad de plantear propuestas novedosas que cuantitativamente y cualitativamente mejoren el rendimiento actual. Como los estándares están aún bajo revisión por el ITU y el ATM FORUM desarrollar herramientas es trascendental para el análisis del control de congestión, del rendimiento genérico de la red y de la correctitud de los protocolos para establecer las rutas de propagación de celdas. Este Trabajo se desarrollo como un modelo computacional y no como un prototipo completo, por ser este último muy costoso y poco flexible.El diseño propuesto puede ser implantado de forma distribuida pero su implantación actual es centralizada. Con esta plataforma es posible estudiar redes BISDN/ATM contemplando sólo a alto nivel el envio, recepción y las operaciones de manejo de colas y red que involucren intercambiar celdas entre los conmutadores. Corre bajo dos tipos de arquitectura: ambientes microprocesadores con interfaz windows y ambientes bajo UNIX.

Palabras Claves: ATM, Redes de banda ancha, BISDN, Redes.

1. Introducción.

Durante la década de los noventa se ha iniciado lo que podemos llamar la segunda revolución de las Telecomunicaciones. Los cambios en el mundo, de la Informática y las Telecomunicaciones, la **Telemática**, han sucedido tan rápidamente que es difícil creer que algo mas importante pase en el resto de la década, algunos analíticos llegan a indicar que la importancia de los nuevos inventos cambiará completamente la forma de vida tradicional y que su impacto en esta última será más relevante que el que originó la máquina de vapor y la famosa revolución industrial.

La integración entre los mundos de la Informática y las Telecomunicaciones han transformado a la tan mencionada **Super Autopista de la Información** en una realidad que ha impulsado el interés por integrar a los Datos, la Voz, el Audio, el Vídeo y la Imagen, en una red internacional, a través de equipos estándares, rompiendo con la barrera de las configuraciones heterogéneas y llegar a un acuerdo internacional para lograr el intercambio de información de los tipos mencionados. Esta red no es otra que la Red Digital de Servicios Integrados de Banda Ancha (BISDN) en una implantación basada en el Modo de Transferencia Asíncrono (ATM). Esta última es tal vez el modelo más evolucionado de una

generación de desarrollos paulatinos en las redes en general que se inició con el Teléfono, siguió con las Redes Conmutadas Analógicas , paso por las primeras Redes de Teleproceso, ancestras estas de nuestras Redes de Area Local de Datos actuales, hasta llegar a las recientes propuestas donde ya se integran voz y datos como SMDS y DQDB, o a las redes de interconexión rápida como FDDI y FDDI II, o a las técnicas de conmutación rápida de paquetes como Frame Relay y Cell Relay. Es en esta última donde aparece el BISDN/ATM y dada su novedad , potencialidad y a la falta en general de estándares es que es necesario crear herramientas para estudiar su comportamiento. El presente trabajo está organizado en secciones la 2 presenta los antecedentes y un breve marco conceptual, la 3 el diseño e implantación de la plataforma y una explicación de su funcionamiento la 4 los resultados y finalmente la 5 las conclusiones.

2. Antecedentes y Marco Conceptual.

El área de simulación de redes por software ha sido una muy activa desde 1980, pero generalmente limitada al desarrollo sobre monoprocesadores. A finales de los años ochenta, Mouftah y Krause [Mouf 88] estudiaron la simulación de controladores de comunicaciones con tráfico mixto, pero no exploraron el control de congestión ni sus políticas, así como tampoco presentaron valores de rendimiento o performance.

En otro trabajo de Kurose y Mouftah [Kuro88] se proponen algunos algoritmos distribuidos sobre procesamiento paralelo para obtener mejores tiempos de procesamiento (CPU Time) para la simulación.Otra propuesta es hecha por Righter y Walrand [Righ90] al señalar que para reducir el tiempo de simulación es necesario asignar más recursos a las tareas.

En 1985, Lubachevsky y Ramakrishnan [Luba85] reportaron un simulador de red desarrollado sobre un sistema multiprocesador de memoria compartida , pero era un esquema basado en tiempo y no en eventos como lo requiere BISDN. En 1988 Baker, Hauser y Thoet [Bake88] aplicaron la idea de simulación distribuida a redes de radio.En 1990 Mouftah y Sturgeon [Mouft90] utilizaron el algoritmo de tiempo virtual propuesto por Jefferson sobre una red de estaciones de trabajo SUN para la simulación distribuida de redes de comunicación en general. Leung en 1990 [Leun90] describe la arquitectura de software que emula sobre monoprocesadores a una red conmutada de paquetes con voz y vídeo, también existe un trabajo de Rayfiel y Silverman [Rayf88] en simulación de redes BISDN, pero no es realmente concluyente.

Ultimamente han aparecido una infinidad en el área de simulación de redes BISDN a nivel de ATM tanto para ser desarrollados sobre plataformas soportadas por ambientes centralizados monoprocesadoras como para plataformas distribuidas sobre redes, en esta última categoría es interesante citar a un trabajo realizado en la Universidad de Brown por Chai y Ghosh [Chai93] que describe una herramienta para modelar y simular redes BISDN a nivel de capa ATM sobre una red de estaciones de trabajo configuradas como un procesador paralelo débilmente acoplado.

BISDN es un modelo referencial para las redes digitales de servicios integrados de banda ancha compuesto por un conjunto de recomendaciones de la CCITT (Comité Consultivo Internacional Telegráfico y Telefónico) actualmente ITU (Unión de telecomunicaciones Internacionales). Al igual que OSI utiliza el concepto de capas. BISDN es un modelo tridimensional dividido en planos (plano de usuario, plano de manejo y plano de control) y capas (capa física, capa ATM , capa de adaptación a ATM y las capas superiores), a diferencia del modelo OSI los planos pueden tener una comunicación directa con las capas[McDy95]. El siguiente trabajo está basado en las características de la capa ATM (Modo de Transferencia Asíncrono) del modelo de referencia BISDN.

El Modo de Transferencia Asíncrono (ATM) es una técnica de transmisión, conmutación y multiplexación, en la cual toda la información es llevada a través de la red en paquetes pequeños de longitud fija de 53 bytes (5 corresponden al encabezado y 48 a la carga efectiva) llamadas celdas ATM. Es un estándar de la tecnología *cell-relay* que toma ventaja del surgimiento de la fibra óptica como medio de transmisión confiable (sólo para las exigencias del modelo BISDN se estipula que ATM trabaje bajo fibra óptica, pudiendo este trabajar bajo cualquier otro medio de transmisión) y del empleo de las celdas que facilitan una mejor multiplexación estadística de la información, una mayor facilidad en la implementación de los conmutadores y una reducción en el almacenamiento interno de los conmutadores. [MILL94]. ATM combina la simplicidad y rápidez ofrecida por la conmutación de circuitos y la flexibilidad de la conmutación de paquetes que va a permitir a esta técnica adaptarse a todas las velocidades. Esta provee un formato común para transmitir vídeo, voz, datos , imágen y sonido; además, garantiza independencia temporal que logra una separación entre el reloj de la fuente y el de la red, así como una independencia semántica ya que para ATM es transparente la información que en la red viaja. Esto permite sólo la detección de errores en la etiqueta (celdas mal direccionadas, perdidas de celdas, etc) [SIMM94]. La multiplexación se realiza colocando las celdas de distintos orígenes dentro de las celdas vacías que fluyen constantemente entre los nodos de la red, si no existe una petición de ancho de banda por ningún servicio, entonces viajan entre los nodos celdas vacías. (Fig 1)

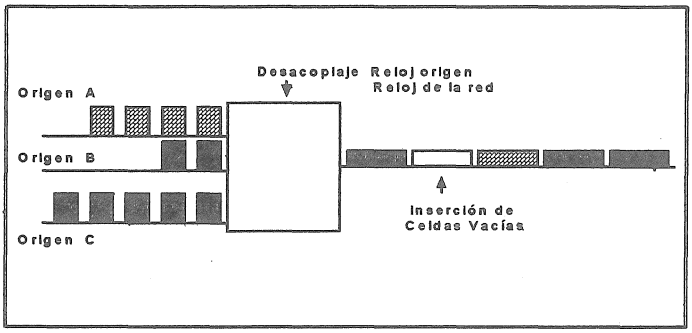


FIG 1.-Multiplexación en ATM

Con la multiplexación en ATM se trata de aprovechar mejor un canal único y la forma en como se comparte este entre varios nodos de la red que lo emplee. Para transmitir información entre nodos, se establece un circuito virtual, el mismo se puede multiplexar para varias aplicaciones, porque dichas celdas se multiplexan estadísticamente, es decir utilizando multiplexación por división de tiempos y sin el empleo de un reloj que las sincronice, ya que la sincronización viene dada por la frontera de bit de cada celda.

Un canal asocia una porción de la capacidad usable a un servicio. ATM redefine este concepto fundamental. Un Canal Virtual (VC) es un trozo de conexión entre dos nodos vecinos. Un trozo de canal virtual se inicia o enlaza con la asignación o supresión del valor del identificador del canal virtual (VCI) contenido en el campo de dirección del encabezado de la celda [BELL91]. Un Camino Virtual (VP) se describe como un conjunto de canales virtuales que se encuentran agrupados dentro de una especie de tubo . En un VP todos los trozos de VC de entrada son los mismos en el extremo de salida. Un VP provee transferencia unidireccional de celdas entre conmutadores, con el mismo identificador de camino virtual (VPI) contenido en el campo de dirección del encabezado de la celda[BELL91] [McDy95].

En ATM existen dos tipos de conmutadores que trabajan dependiendo de la etiqueta que van a cambiar, si la etiqueta a modificar es sólo el VPI (traducir un valor de VPI entrante en un valor de VPI saliente) se dice que estamos en presencia de un conmutador de VP, si por el contrario modifican una combinación VPI/ VCI se dice entonces que es un conmutador VP/VC.(Ver Fig 2) El conmutador VP/VC permite al usuario el establecimiento de llamadas mientras que el conmutador VP sólo se encarga de los comandos de gestión de la red. En el trabajo sólo se simula el comportamiento de los conmutadores VP.

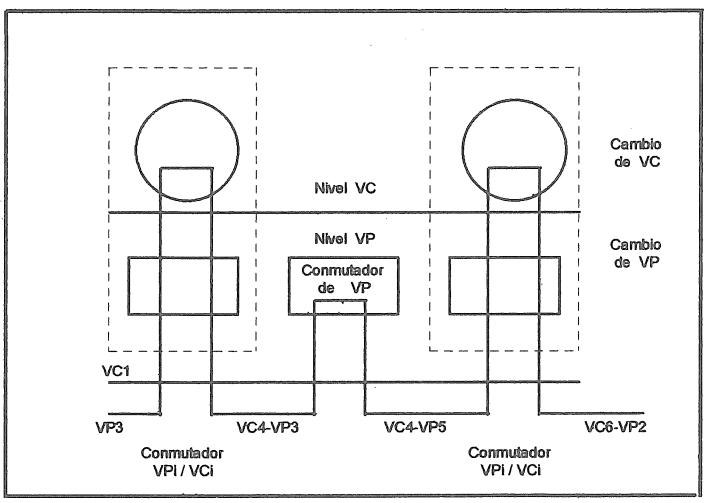


FIG 2.- Conmutadores VP/VC y VP

3. Diseño e implantación de la plataforma:

La plataforma plantea una forma de resolver el problema de simular una red ATM a nivel de envío y recepción de celdas. La misma es una solución centralizada la cúal puede ser llevada a un entorno distribuido, haciendo una generalización de ésta en dicho ambiente.

La plataforma se divide en el módulo del servidor y el módulo de Conmutación.

Cada uno de estos trabaja de manera disjunta, llevando a cabo un conjunto de funciones específicas, donde el módulo del servidor se encarga en líneas generales de la gestión de la red mientras que el módulo de Conmutación realiza las funciones propias de un nodo de conmutación dentro de la misma.

Las funciones del módulo servidor se subdividen en: Funciones de inicialización y funciones de temporización y control.

Las funciones de inicialización se resumen en: Generación de todas las direcciones de cada conmutador en la red e intercambio con sus conmutadores vecinos, envío a cada conmutador de sus tiempos de procesamiento, generación de los retardos de transmisión, capacidad, longitud, retardo de las celdas y tiempos de vida de cada uno de los enlaces que existen entre los conmutadores, calculo de los caminos y canales virtuales entre cada par de conmutadores utilizando el algoritmos de enrutamiento (el trabajo utiliza el algoritmo de camino minimo), distribución de estos a cada uno de los conmutadores, definición de las conexiones de envío y recepción de los conmutadores.

Por otra parte las funciones de temporización y control se reducen a: manejo del reloj global del sistema, restauración de las conexiones en caso de fallas de enlaces (actualizando las tablas de enrutamiento de los nodos adyacentes a ese enlace) y reestructuración de los caminos y canales virtuales en caso de congestión. La Figura 3 muestra la relación entre el modulo servidor y conmutador.

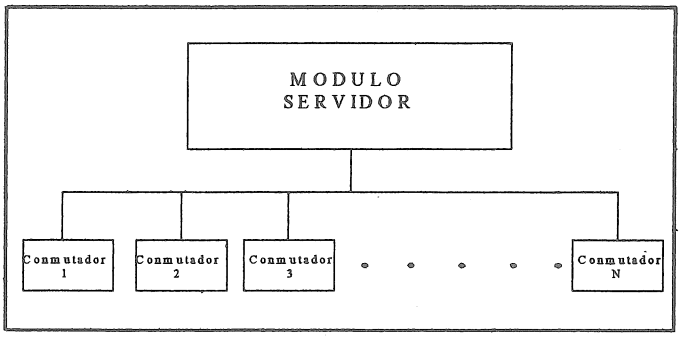


FIG 3. - Relación entre el módulo Servidor y los conmutadores

Las funciones del módulo de conmutación son: simulación del comportamiento del tráfico de la red, ejecución de las operaciones de multiplexación y demultiplexación para la compartición de líneas de comunicación, enrutamiento de los mensajes, establecimiento de la comunicación entre los nodos de la red, selección del modo de operación del conmutador (conmutador VP o VP/VC), recepción de las

celdas de entrada, envío de las celdas de salida y por último reencolaje, descarte o transmisión de la celda, por un enlace determinado.(Fig 4)

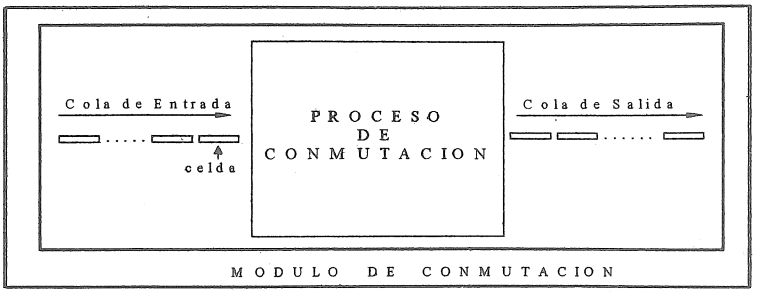


FIG 4.- Módulo de conmutación

La plataforma consta de un reloj global y un conjunto de eventos que están asociados a las funciones del módulo servidor y del módulo de conmutación. Los eventos son descritos a continuación:

- Evento de Inicialización.

Inicialmente se generan el número de celdas a utilizar por cada conexión en un instante de tiempo determinado dentro de la simulación, éste proceso es realizado por el módulo del servidor y consiste también en repartir toda la carga de trabajo (que se extrae de los archivos de entrada donde está almacenada además del número de celdas a transmitir, toda la información referente a los conmutadores de la red y las conexiones establecidas en ésta) entre todos los módulos de conmutación (asociados a los conmutadores de la red a simular) y luego asigna a cada proceso una identificación unívoca.

Se procederá a ejecutar seguidamente el módulo del servidor en cuestión, quien realizará el proceso de inicialización del sistema seguido del proceso de temporización y control, que dará inicio a las ejecuciones concurrentes de los módulos de conmutación asociados a cada conmutador de la red, que previamente fueron definidos para operar como conmutadores VP. Se procede a generar los tiempos de propagación (Longitud del enlace/ Velocidad de propagación de la fibra óptica (Velocidad de la luz)), capacidad (Velocidad de transmisión (155,52 Mbps que es el estánda rque mayormente se emplea en las redes ATM.) / Tamaño de la celda (53 * 8 bits) = 366792 celdas), longitud (longitud física del medio (Km)) y tiempo de vida de cada uno de los enlaces que son entrada de la plataforma.

Luego el módulo del servidor ejecuta el programa que calcula los caminos y canales virtuales, según un algoritmo de enrutamiento preestablecido que en este trabajo se utilizó el algoritmo de camino minimo [TAN 91]; y distribuye dicha información en las tablas de enrutamiento de cada uno de los conmutadores . Además genera en cada uno de estos una tabla que indica los enlaces de envío y recepción de éste. En vista de que la simulación es implantada en un ambiente centralizado, no se utiliza una tabla de enrutamiento en cada conmutador, que contiene el camino entre un conmutador y cualquier otro conmutador de la red. A consecuencia de esto el sistema mismo se encarga de

seleccionar a partir del camino mínimo , el enrutamiento y lo coloca de manera implícita en la tabla virtual a medida que se genera una conexión, creando una entrada que pertenece a dicho camino para esa conexión en la tabla del conmutador correspondiente. Por último se inicializa el reloj global del sistema con el primer tiempo de simulación que se establezca en la entrada para alguna conexión. A vontinuación se describen los eventos que manejan al sistema.

- Evento de generación de celdas.

Consiste en generar tantas celdas como lo indique el ancho de banda requerido para la conexión, y colocarlas en las colas de salidas respectivas si estas no estan llenas, además se le coloca el reloj de la celda con un valor igual al último tiempo de reloj de la última celda a generar en la misma conexión, y se le coloca el valor del VPI igual al que indique la tabla virtual para esa conexión. Además, se cambia el valor del reloj global del sistema por el mínimo de todos los relojes locales de los conmutadores, esto si alguno de ellos es menor que dicho reloj global.

- Evento de envío de las celdas por los enlaces.

El módulo de conmutación en cuestión se encarga de enviar las celdas por los enlaces correspondientes tomando las que se encuentran en las colas de salida(con reloj de celda igual al reloj global) y chequeando si no existe falla o congestión en el enlace por el cúal esta debe transitar, de lo contrario procede a reencolar o descartar a la celda según sea el caso, tomando en cuenta previamente el tiempo de permanencia máxima, el retardo permitido y el tiempo de vida de la celda para incrementar así el número de descartes asociado a dicho parámetro.

- Evento de procesamiento de las celdas en el conmutador.

Se toman las celdas que esten en las colas de entradas de los conmutadores (con reloj de celda igual al reloj global) y se colocan en las colas de salida correspondiente a su enlace de salida, según alguna política (en el trabajo se utiliza la política FIFO que es una de las mayormente conocidas y utilizadas de esta forma no se toma en cuenta el bit de prioridad de las celdas), seguidamente se procede a actualizar el reloj de cada celda dependiendo del valor del reloj global del sistema, el tiempo de procesamiento del conmutador,el tiempo de transmisión del enlace desde donde proviene,el tiempo de vida de la celda y el tiempo de permanencia en cola (que es el tiempo de permanencia en el conmutador). Además es también actualizado el identificador de VPI de la celda con el indicado en la tabla de enrutamiento y el reloj local del conmutador incrementándose con el menor valor del reloj de celda de todas las celdas que esten en su cola de entrada y de salida.

- Evento de Actualización.

Consiste en actualizar el reloj global del sistema con el minimo de los relojes de las celdas que se encuentran en las colas de los conmutadores. Además actualiza el tiempo de vida, permanencia en el sistema y en el conmutador de las celdas; y el tiempo de vida de los enlaces.

Existen distintos parámetros evaluados en las redes ATM [MILL96]. Los parámetros que se tomaron en cuenta para la simulación son:

TIEMPO DE PROPAGACION: Depende de la distancia entre dos puntos finales y se define como el tiempo que tarda una celda en atravesar un enlace entre un par de conmutadores.

NUMERO DE CELDAS DESCARTADAS POR LONGITUD DE LA COLA: Es un valor que indica el número de celdas que se descartan en la simulación por exceso en las capacidades tanto de las colas de entrada como de las colas de salida.

TIEMPO DE VIDA DE LA CELDA: Es el tiempo máximo permitido que la celda está activa en la simulación. Este valor existe desde que la celda es generada hasta que esta es procesada en el destino (si esta no es descartada). Su actualización se realiza decrementando su valor cuando se actualiza el paso de tiempo. Este parámetro permite medir el número de celdas descartadas por vencimiento del tiempo de vida.

TIEMPO DE PERMANENCIA DE LAS CELDAS EN EL SISTEMA: Es el tiempo máximo permitido en el cual la celda está en el sistema y sólo se decrementa cuando esta es reencolada, es decir, es el retardo máximo permitido para que la celda se reencole en el sistema.Este parámetro permite medir el número de celdas descartadas por vencimiento del tiempo de permanencia de la celda en el sistema.

TIEMPO DE PERMANENCIA DE LAS CELDAS EN EL CONMUTADOR: Es el tiempo máximo permitido que la celda puede estar en el conmutador, este abarca el tiempo que permanece la celda en las colas de entrada/salida del conmutador y el tiempo que toma este en procesarla. Este tiempo puede variar entre conmutadores. Este parámetro permite medir el número de celdas descartadas por vencimiento del tiempo de permanencia en el conmutador.

TIEMPO DE VIDA DEL ENLACE: Es el tiempo de duración que tiene el enlace mientras se ejecute la simulación y difiere para todos los enlaces que constituyen la red; permite además medir el número de celdas descartadas por falla de enlace.

NUMERO DE CELDAS PROCESADAS: Es el número de celdas que logran alcanzar con éxito el destino final.

IMPLANTACION:

La implantación se realizó en C ++ sobre un computador PC/AT 486 SX33 (Compatible IBM), con 8 Megabytes de memoria RAM, un disco duro 203 Mb, una unidad de disco de 3.5 de 1.44 Megabytes y un monitor VGA. El software que se empleó para realizar la implantación consta de un Sistema Operativo 6.20, Windows version 3.1 y Visual Basic 3.0 para la realización de la interfaz. Dicha interfaz provee al usuario la facilidad de obtener en cada uno de los ambientes de trabajo una ayuda en linea, además está desarrollada de una manera amigable en el entorno de windows .También se adapto para plataforma UNIX y fué probado en una Challenge de Silicon Graphics sin la interfaz gráfica.

De manera general el programa se estructura de la siguiente forma:

Evento de inicialización

CICLO

Evento de generación de las celdas
Evento de envío de las celdas por los enlaces
Evento de procesamiento de las celdas en el conmutador
Evento de actualización del reloj global
HASTA (No existan celdas a generar o tratar en el sistema)

En la simulación el reloj global es llamado ***tiempo actual*** que garantiza que todos los eventos ocurran en un mismo instante de tiempo y se lleven a cabo como si se realizaran de manera simultánea. El mínimo de todos los relojes locales de los conmutadores es llamado ***tiempo tentativo*** que permite controlar el solapamiento de las conexiones e implementa los relojes locales de los conmutadores de una manera glogal. Todos los procedimientos y variables forman parte del programa que implanta el trabajo[MILL96].

Son dos las estructuras de datos principales utilizadas, la primera es una lista donde se encuentra almacenada toda la información necesaria para cada uno de los conmutadores de la red (que representan a cada uno de los módulos de conmutación de la red a simular), a su vez en cada uno de ellos existen tres listas que representan las celdas que están en su cola de entrada, las celdas que están en su cola de salida y la tabla virtual, donde esta última contiene para cada instante de tiempo las conexiones que se generan , el número de celdas que se desean enviar y el destino según el origen determinado para la conexión . La segunda estructura guarda información concerniente a los enlaces de la red, que permitirá conocer el estado de estos en cualquier instante de la simulación.

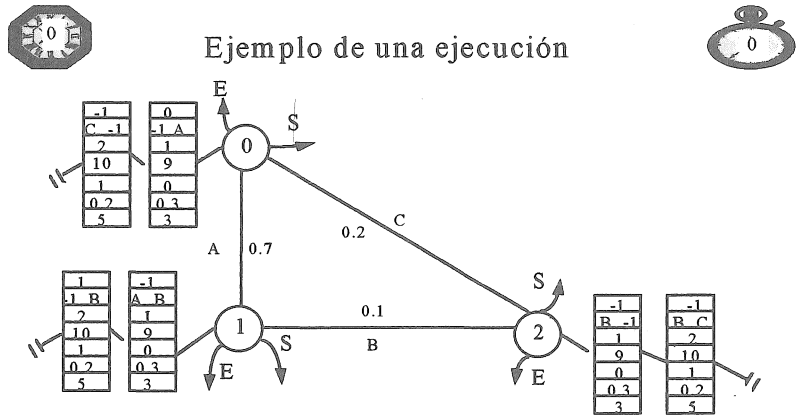


FIG 5.- Ejemplo de una ejecución.

La Figura 5 representa de forma ilustrativa un ejemplo para una red de 3 nodos, conectados bajo una topología anillo; donde se nota en la fase de inicialización, el estado de las tablas virtuales de

cada conmutador para dos conexiones: entre los conmutadores 0 -1-2 y 1-2-0. Además se observa en la parte superior izquierda el reloj global del sistema y en la derecha el reloj tentativo.

A continuación se observa una de las ventanas principales de la plataforma, que despliega uno de los ejemplos que contiene en el menú de opciones (Fig 6):

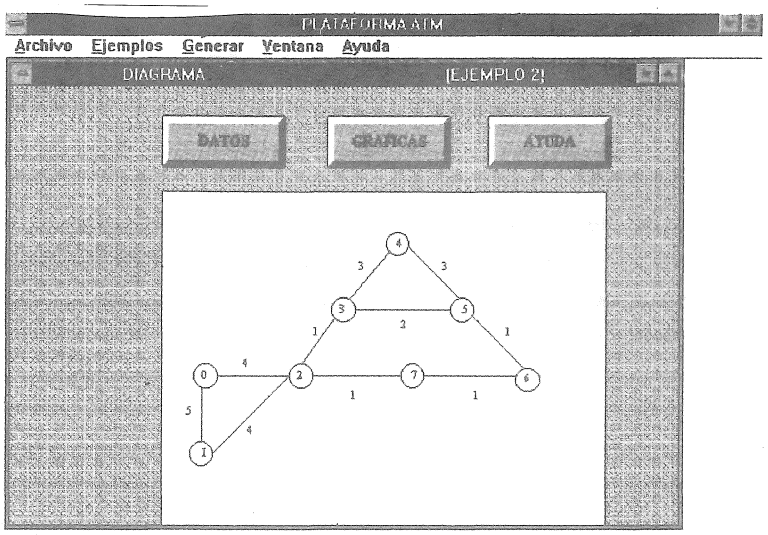


FIG 6.- Ventana principal de la plataforma implementada.

4. Resultados:

Para la evaluación de los resultados fue necesario la emisión de un conjunto de ejemplos que permitieron probar y hacer un estudio comparativo de diversas redes, que varian en cuanto al número de nodos, número de tiempos a simular, anchos de bandas requeridos, carga de trabajo (medida en número de celdas) , número de conexiones y tiempo de procesamiento de los conmutadores. Se analizaron para cada ejecución todos los parámetros que fueron referidos en el diseño que permitieron llegar a conclusiones sobre los requerimientos, alcances y fallas de las redes en estudio.Una de las redes estudiadas está conformada por 13 nodos interconectados a través de topologías con estructuras de anillos y buses. El tamaño de los enlaces varia entre 2 y 4 Kilometros de longitud y los tiempos de vida de estos son de 2.5. El tiempo de vida de las celdas y el de permanencia de estas en el sistema es de 0.005. El tiempo de permanencia en el conmutador 0.0025. El tiempo de procesamiento de los conmutadores es de 0.02, 0.01 y 0.1.Se realizaron para este mismo ejemplo tres ejecuciones donde sólo

se varió el tamaño de las colas de entrada/salida (50, 100 y200celdas).La figura muestra la red descrita.

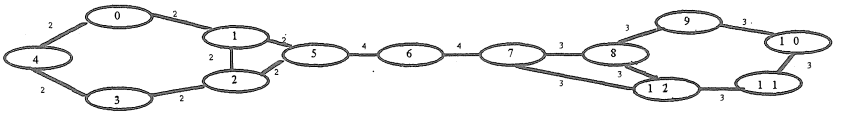


FIG 7.- Topología de la red

Para esta red se establecieron tres tiempos a simular en el tiempo 0 existen 5 conexiones con anchos de bandas entre 0.84 y 3 Mbps, con un máximo de 200 celdas para una conexión, para el tiempo 1 se tienen 3 conexiones con anchos de bandas que varían entre 0.25 y 1 Mbps con un máximo de 30 celdas para una conexión y por último para el tiempo 2 se tienen 4 conexiones con anchos de bandas entre 0.42 y 2 Mbps con un máximo de 180 celdas para una conexión. Se generaron un conjunto de gráficas para las distintas ejecuciones correspondientes a los ocho parámetros antes mencionados. En la figura 8 se muestra una de las gráficas que se generó para la red antes mostrada (con una capacidad en las colas de los conmutadores de 50 celdas), ésta refleja el número de celdas descartadas por tamaño de cola de salida llena. En general, el tamaño de las colas de los conmutadores de la red constituye un factor crítico, se observó que a medida que se incrementa el tamaño de las colas, menor es el número de celdas descartadas por su capacidad. El incremento en el número de celdas descartadas por tamaño de colas también está relacionado con el uso que tengan los enlaces (mientras más transitado está un enlace los tamaños de colas de los conmutadores que lo conforman crece). A medida que el tamaño de las colas se incrementa mayor es el descarte por tiempo de permanencia en los conmutadores, en el sistema y el tiempo de vida de las celdas, ya que mucho más tiempo pueden quedarse las celdas dentro de la red.

FIG 8.- Gráfica descarte por cola de salida llena.

El ancho de banda de los enlaces no es un factor que se varía en la simulación. Debido a la gran capacidad que éste presenta, en la mayoría de las ejecuciones realizadas para la prueba de la plataforma no hubo reencolaje de celdas. Además, se observó que en un determinado paso de tiempo ocurren la mayor parte de los descartes, este pudiera representar el instante de tiempo en el cúal la red tiende a congestionarse. El tiempo de permanencia de las celdas en los conmutadores es un factor determinante para medir el tráfico de la red, ya que las celdas pueden permanecer mas tiempo dentro de estos y así decrementar el número de celdas descartadas por el vencimiento de dicho tiempo. De manera general, cada uno de los parámetros evaluados, juegan un papel importante en el comportamiento de la red, y el mejoramiento de ésta depende de cuan minucioso sea el análisis de los mismos, este caso se presenta sólo como ejemplo para mostrar las potencialidades de la plataforma.

5.Conclusiones.

La plataforma presentada en este trabajo ayudó a medir ciertos parámetros presentes en las redes ATM, comprobándose la gran importancia de desarrollar prototipos que ayuden a la puesta en marcha de este tipo de redes. Se comprendió en la elaboración del mismo, la importancia de estudiar el comportamiento a priori de redes ATM, a través del uso de una plataforma. Además la plataforma permite: llevar un seguimiento del comportamiento de una red preestablecida, producir información de monitoreo de la red, visualizar los resultados de una manera gráfica amigable, brindar facilidades de ayuda en linea al navegar en esta, etc. Finalmente, este trabajo forma una parte importante que junto con el desarrollo de las capas SONET/SDH, AAL y capas superiores, conformar así una gran plataforma que se denominará BISDN, esto se puede lograr gracias a la modularización con la que este fue diseñado.

REFERENCIAS

[BAKE88] Baker, D. J.,Hauser, J.P., Thoet, "A designation and Prototyping testbed for radio comm works" IEEE journal Selected Areas in comm. 1988 pp197-210

[BELL91] BELLEC, Pierre-Yves. "ATM". France Telecom. France 1991. pp 54.

[CHAI93] CHAI, Arthur and GHOSH, Sumit. "Distributed Modeling and Simulation of a BISDN network". Computer, pp 37-51Sep 1993.

[ICL 94] Interconnect Comunnications LTD and European SMDS Interest Group. "SMDS HANDBOOK" Publishing Quorum Technical Services Ltd. United Kingdom. March 1994, pp 98.

[KURO88] Kurose, J,K., Mouftah, H.T., "Computer aided analysis and design of computer networks" IEEE Sel in Comm vol 6, no. 1 jan 1988,

[LEUN90] Leung W. H. F., "A software Architecture for supporting Multimedia conferencing in packet networks"IEEE J. Selected Areas in Comm. 1990,pp380-390.

[LUBA85] Lubachevsky B., y RamaKrishnan K., "Parallel evaluation of a network using shared memory modeling tools and techniques for performance"North Holland, Amsterdam 1985

[McDy95] McDYSAN, David E and SPOHN. Darren L. "ATM. Theory and application". McGraw-Hill, INC. United States of America. April 1995. pp 636.

[MILL94] MILLER, Arthur. "From here to ATM". IEEE Spectrum, pp 20-24.June 94

[MILL96] MILLAN, Yrielis y RIVAS Rachele. "Desarrollo de una plataforma de para el estudio de redes ATM". U.C.V. Caracas, Venezuela. Febrero 1996 .

[MOUF88] Mouftah, Sturgeon "Distributed evaluation for communication networks"IEEE J. Selected areas in comm. vol 8, No. 9 Dec. 1990 pp1723-1730

[RIGH90] Righter R., Walrand J.C. Distributed simulation event systems" Proc IEEE vol 77 No.1.1990

[SIMM94] SIMMONS, Jane M and GALLAGER, Robert G. "Design of Error Detection Scheme for Class C Service in ATM". IEEE/ACM Transactions oNetworking,vol 2, Nº 1. February 1994.

[TAN91] TANEMBAUM, Andrew S. "Redes de Ordenadores". Editorial Prentice Hall. Segunda edición. Mexico 1991. pp 759.

A GENERIC ARCHITECTURE FOR USE-CASE BASED OBJECT-ORIENTED DESIGN [1]

F. Losavio, A. Matteo
Centro de Ingeniería de Software Y Sistemas ISYS,
Facultad de Ciencias, Universidad Central de Venezuela,
Apdo. 47567, Caracas, Venezuela
flosavio@conicit.ve / @anubis.ciens.ucv.ve
amatteo@conicit.ve / @anubis.ciens.ucv.ve

Abstract

A generic framework is defined for describing a reusable architecture of a system, according to a use-case based design and it can be used for any system developed according to Jacobson's OOSE method. The layered architecture obtained is constituted by three main levels: problem domain, control and interface level, where the interface and control levels are structured as a function of the system's actors. Particularly, for the interface level, interface agents are also modelled with frameworks. Notice that, with respect to the interface level, the general framework provides a greater level of genericity, in the sense that it can be adapted to the framework implementing the interface agents, according to different multiagent models, such as MVC or PAC. Moreover, different frameworks are used for implementing each level and then plugged into the generic framework.

Keywords: use-case, object-oriented design, user-interface design, framework, design patterns, multiagent model, interface agent, PAC model

1. INTRODUCTION

The goal of this work is to derive a generic framework for object-oriented (O-O) design from the system's actors functionalities, captured by the use-case concept, according to a scenario-based [Car 95] point of view. A framework is defined as a reusable semifinished architecture for various application domains [Pre 94]. It is constituted by a set of classes, that may constitute several design patterns, conceived for working together, representing a generic subsystem that can be instantiated. The instantiation is acheived developing subclasses from the public abstract classes of the framework, called "hot spots" [Pre 94], which are seen as predefined places where application specific parts are composed. A developer using a framework must know the hot spots for a given problem and know how to adapt them to the application's needs. The MVC (Model-View-Controller) model [Gol 84] for building graphical user-interfaces (GUI) is considered one of the first known frameworks [KP 88]. Use-cases are identified in the OOSE (Object Oriented Software Engineering) requirements model [Jac&Al 92] and the framework is established on the basis of a general schema [LM 95] for organizing entity, control and interface objects, considering a given "human" actor. This framework integrates other frameworks, considering three layers or levels, for achieving a complete system design: *problem domain level*, *control level* and *graphical user-interface (GUI) level*. The problem domain level can be modelled using any existing O-O methods or a combination of them, appearing now as an incouraging trend for developing and using object-oriented methods. The interface between this level and the control level is established through the Facade design pattern [Gam&Al 95]. The framework for the control level is built under the basis of generating control classes for each actor and their respective GUI interface classes. The GUI level is derived from the interface classes applying also a generic framework for GUI design based on multiagent models [LM 96].

This paper is structured into three main sections, besides this introduction and the conclusion. Section 2 is dedicated to a brief presentation of the OOSE method, focusing on the heuristics and strategies followed to

[1]This research is supported by the New Technology Program of the BID-CONICIT, the CONICIT MOODe Project and the ISYS center OOMGRIN Project.

characterize interface and control objects in the analysis model. Section 3 presents the schema obtained from the interface objects of the OOSE analysis model. Section 4 is focused on the generic framework for system's design, goal of this work.

2. THE OOSE (OBJECT ORIENTED SOFTWARE ENGINEERING) METHOD

The OOSE method [Jac&Al 92] encompasses the application development in terms of three main processes: Analysis, Construction and Testing, each maintaining their own models. Figure 1 below shows the OOSE method through the use-case modelling approach. The analysis step, concerns the obtention of the Requirements and Analysis Model. The use-case driven approach is important during this step, considering the requirements expressed in terms of actors, representing the people, systems and devices that will exchange information with the application. Use-cases represent events in terms of "what happens when" scenarios for each actor. They provide a common ground for developers and their customers to acheive a shared vision of the application, early in the development process, when inconsistencies are easy and inexpensive to correct. The terms use-case and functionality will be used indistinctly through this work. The use-cases identified during requirements analysis continue through the whole OOSE life cycle, driving each subsequent phase. In the Analysis Model, three types of objects for each use-case are identified: *interface*, *entity* and *control*. Interface objects (for example graphical user-interface widgets or complex windows), represent objects that depend on and change with the application's interface environment. Entity objects represent persistent data of the application, expressing its semantics and control objects contain rules not directly dependent on data or interface. We will be interested here in control and interface objects.

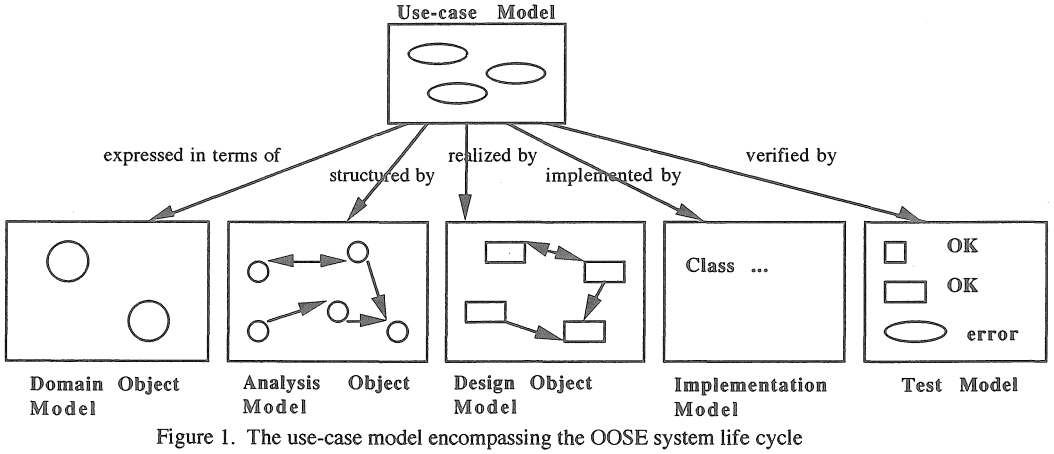


Figure 1. The use-case model encompassing the OOSE system life cycle

2.1 Control objects of the Analysis Model.

According to the OOSE approach, entity and interface objects should have been identified before the control objects. These are then used as a "cement" to relate or link the other objects, in order to constitute a complete use-case. Control objects are non-persistent, in the sense that they only exist during the use-case execution. In a use-case, the sequence of actions may follow different alternatives, constituting a *scenario of events*. Control objects allow also communication among the other objects, connecting different scenarios.

Some heuristics for identifying control objects are [Jac&Al 92]:

- They are determined from the use-cases. At first, a control object can be assigned to each concrete or abstract use-case, where the interface and entity objects are implicit and have already been identified. Then,

control objects express the behaviour that has not been captured in the other objects. There is no need to model control objects, when there are no more features to be captured through them. If, on the contrary, a complex behaviour has to be modelled, the functionalities can be subdivided into several simple control objects, with specific and limited tasks, easy to describe.
- When a control object is related to different actors, its behaviour depends on the actor and it shquld be decomposed into different control objects, one for each actor. This decomposition favors extension and maintenance because modifications are induced by the actors. Hence, system modifications may be easily isolated if a control aspect is associated to a unique actor. In general, the kind of functionalities expressed by a control object are transaction oriented or particular control sequences of one or several use-cases.

2.2 Interface objects of the Analysis Model.

All the functionalities specified in the use-case descriptions depending directly from the environment exterior to the system, are expressed by the interface objects. The actors communicate with the system through these objects. The role of an interface object is to translate the actor actions on the system into system events involving the actor and its queries. Interface objects describe bidirectional communications between the system and its users.
Interface objects are identified according to [Jac&al 94]:

- Through the interface descriptions of the Requirements Model.
- Through the actor role.
- Through the use-case description, extracting those functionalities specific to the interface

Changes on the system interface are in general frequent and expansive and it is important to have criteria to manage them accordingly. Interface objects facilitate extension of the system interface, since the impact of modifications are concentrated only on these objects.

According to [Jac&Al 92], two kinds of interfaces may be modelled for a problem solution: interfaces with other systems and human user interfaces. For the first kind of interfaces, the communication is described in terms of *protocols*. For the second kind, GUI are used for communication between the user and the system. Several systems such as X-Window[2], News [Ste 87], NexTStep [Gar 93] and Windows[3], among many others, and class libraries written in different languages such as C++ [Str 93] and Smalltalk [Gol 84], are widley used tools, constituting also standards for developing systems with large GUI components. Different levels of granularity characterize the modelling of the interface objects. In general, a window is considered an interface object. In order to model each window structure, the gadgets or widgets of the tooolkit corresponding to the particular windows manager system are also considered interface objects, constituting the fine grain levels. A *central interface object* corresponds to a whole window and the other interface objects model the multiwindows toolkit widgets. Interface objects focus on the presentation, but can also process information and hold a certain behaviour. The amount of information and behaviour for each interface object is decided accordingly to each particular case. The same occurs with the other objects. All the use-case functionalities must be distributed into entity, interface and control objects. There are several strategies for assigning these functionalities [HH 89]:

i. *Integrated control*: when the control functionalities are placed into control and entity objects. In this case, the interface objects are not relevant, because they have no functionalities. This structure may be efficient

[2] X-Window is a registered trademark of the Massachusetts Institute of Technology.
[3] Windows are registered trademarks of Microsoft

with respect to execution time, but is not convenient for prototyping because the interface objects have very few functionalities.

ii. *Dialog oriented control*: when most of the dialog control functionalities are placed into interface objects and these model most of the system functionalities. In this case the control objects are not relevant, because they have few functionalities. This structure facilitates prototyping, but increases the interface complexity, due to the fact that events are detected and processed in the same object.

iii. *Mixed control*: when control functionalities are divided into entities and interface objects, allowing, for example, to invoke the dialog from an entity object, where some computations are performed, and reciprocally. In this case, communication facilities are not concentrated on particular objects, not easing modifications.

iv. *Balanced control*: when the control is completely separated form the computing aspects of the entity objects and from the dialog aspects of the interface objects. Then in this case, control objects manage communication between entity and interface objects.

3. A SCHEMA FOR DEFINING ANALYSIS OBJECTS IN OOSE

This schema [LM 95] defines the objects in the OOSE analysis model, with respect to a given "human" actor and the use-cases corresponding the GUI functionalities. It shows the control and interface objects which have to be defined, considering the different use-cases related with a certain actor and establishing communications among these objects. The balanced control strategy described in 2.2, where the control object is an intermediary between the entity and interface objects, is taken into account in order to derive the schema, which is presented in Figure 2. The lines between the objects represent static links. For the definition of this schema, the following strategies are considered:

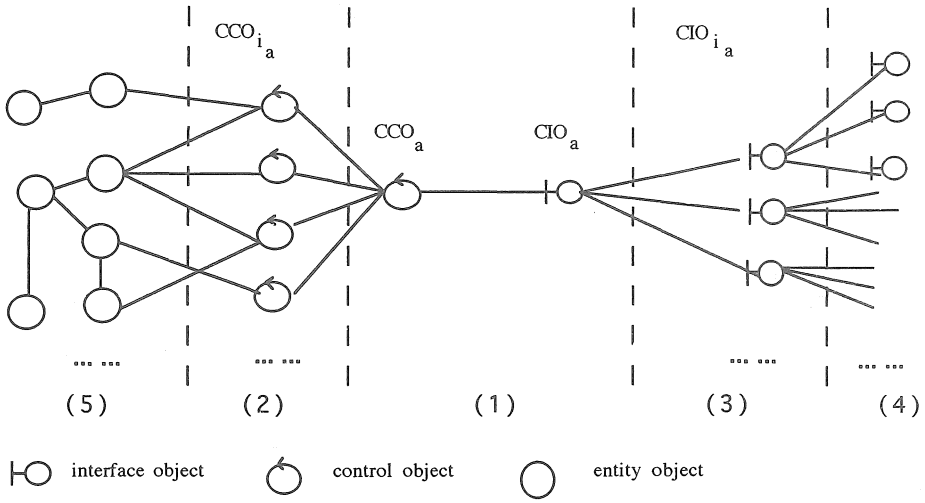
i. Define a central interface object, denoted by CIO_a, associated to the actor, denoted by *a*, corresponding to a window from which an actor can access the corresponding functionality (Level 1 in the schema). Define central interface objects CIO_{i_a}, $1 \le i \le n$, where *n* is the number of use-cases associated to *a*, each one corresponding to a window, from which one or more functionality or use-case can be executed (Level 3 in the schema). Then we have at most n CIO_{i_a}.

ii. Define a *central control object*, denoted by CCO_a(Level 1 in the schema). For each functionality or use-case, define a control object, denoted by CCO_{i_a}, $1 \le i \le n$, where *n* is the number of use-cases associated to a (Level 2 in the schema). The CCO_a expresses the control requirements of the actor. It manages the distribution of controls among the actor's functionalities, on the basis of the balanced control strategy, allowing also a clear separation between the abstraction and the user-interface aspects. The control object CCO_{i_a} associated to an actor facilitates the possble modifications relative to each functionaliy.

iii. Create independent or completely separated entity objects (Level 5 in the schema) from interface objects. This separation means that they do not "know" each other, in the sense that no links (static or dynamic) are established among them. Notice that the balanced control strategy (see 3.2) is also followed in this case.

iv. Locate toolkit widgets or widgets composition, to construct the windows corresponding to the CIO_{i_a}, $1 \le i \le n$ (Level 4 in the schema). These objects are non central interface objects.

The above strategies clearly favor the separation between entity and interface objects, easing the system's decomposition into interface and problem-domain components. Notice that the OOSE approach does not treat explicitly this separation, allowing links between entity and interface objects [Jac&Al 92]. As a consequence of the proposed schema, easy location of extensions or modifications, and the programmers' teamwork during the implementation phase are facilitated [Los&Al 94a], [Los&Al 94b].



(1) Central control and interface objects associated to the actor (CCO_a, CIO_a respectively)
(2) Control objects, one for each use-case associated to the actor (CCO_{i_a}, $1 \le i \le n$)

(3) Central interface objects, associated to the different use-cases of the actor. Each interface object in this level corresponds to a window, allowing the access to one or more functionalities (use-cases) associated to the actor (CIO_{i_a}, $1 \le i \le n$)

(4) Interface objects that are not central, such as widgets or objects composed from widgets
(5) Entity objects implied in the solution of the different functionalities (use-cases) associated to the actor

Figure 2. Schema of objects definition with respect to a given "human" actor

4. A FRAMEWORK FOR USE-CASE BASED DESIGN

A generic framework for object-oriented system design is presented in this section (Figure 7). From the above schema (Figure 2), the traceability of the control and interface objects with respect to a given actor can be followed through the actor's control objects framework (Figure 3) and the actor's window with related use-case windows framework (Figure 6). The entity objects are shown as part of the outside world in Figure 3, as the problem domain component. Figure 4 and 5 show two frameworks for GUI interface agents, corresponding to different multiagent models. One of these framework can be used for designing the GUI component, particularly to construct the framework shown in Figure 6, which can be "plugged" into the control objects framework (figure 3) for obtaining a generic framework for the general architecture of the whole system, shown in Figure 7. In what follows, these frameworks will be discussed in details, to illustrate the complete system design obtained with this approach.

4.1 Framework for control objects associated to an actor

From the above schema, the Central Control Object CCO_a, has been defined, for distributing the user requirements with respect to the different use-cases of a given actor. This object communicates with the GUI level, through the CIO_a , corresponding to an actor window, from where the actor's different GUI functionalities are accessed, inplemented by the Agent framework of Actor *a*. The ControlCalss Actor $_a$ will be activated or created by the SystemTop-Level Control class (see Figure 3), representing the central control of the whole system.

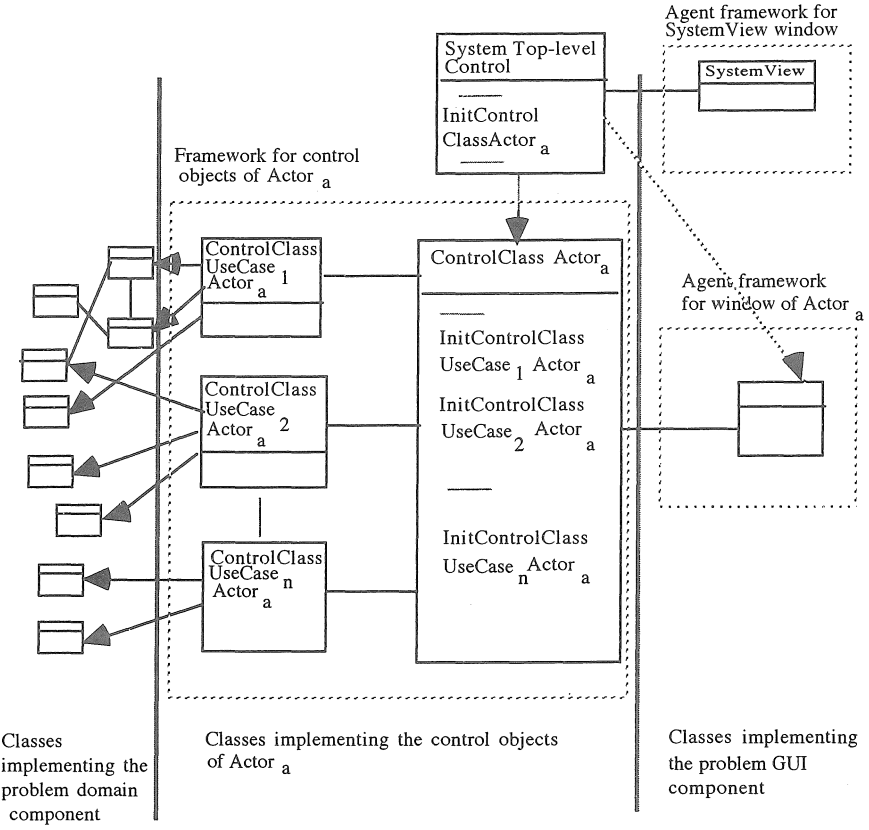


Figure 3. Framework for the control objects of an actor

Moreover, control objects CCO_{i_a}, for each use-case of the actor, interact with CCO_a, who discriminates, at a certain moment, the control objects responsible of the particular functionality required by the actor. Besides, each one of these control objects connects the entity objects required for the solution of the respective functionality. The framework corresponding to the actor's control objects is defined, considering exactly one class for each control object; the relations among these classes are justified by the above discussion. Moreover, the Control Objects framework, denoted by the dashed rectangle in Figure 3, shows also the connections with the exterior world, represented by the classes implementing the entity objects, belonging to the problem domain component, and by the classes implementing the GUI agents of the problem interface component. The ControlClassUseCase$_i$, implement the CCO_{i_a}, $1 \le i \le n$, and corresponds to the FACADE design pattern [Gam&Al 95]. This pattern defines a higher interface level that makes the

subsystem easier to use, structuring a system into subsytems reduces complexity. A common design goal is to minimize the communications and dependences between subsytems. The FACADE object helps to acheive this goal providing a single simplified interface to the more general facilities of a subsystem. The classes implementing the entity objects and corresponding to some functionality, may be seen as subsytems. The FACADE pattern is used when there are many dependences between clients and the implementation classes of an abstraction. The subsystem is decoupled from clients and other subsystems, promoting subsystem independence and portability. Another important usage of FACADE is to define an entry point to each subsystem level. If subsystems are dependent, then the dependencies can be simplified by making them communicate with each other only through their facades. FACADE knows which subsystem classes are responsible for a request and delegates client requests to appropriate subsystem objects. Subsystem classes implement subsystem functionality, handle work assigned by the FACADE object and have no knowledge of the FACADE, keeping no references to it. So, a convenient interface is defined, for solving the functionality related to use-case *i* of actor *a*. In other words, ControlClassUseCase$_i$ $1 \leq i \leq n$, sends messages to the classes corresponding to the entity objects required for the use-case solution. This class is also connected and interacts with the ControlClass Actor$_a$, which is responsible of interacting with the GUI classes related to the actor, discriminating which on of the ControlClass UseCase$_i$ must appear at a given moment, to satisfy user requirements.

4.2 Framework for interface objects associated to an actor

Interface objects can be structured and defined according to each "human" actor, as we have seen in Section 3. These interface objects correspond to the GUI component of the system. In order to define this architecture, the well known MVC or PAC multiagent models can be used, favoring separation between semantic and presentation aspects of the system, enhancing its adaptability. As a consequence, we can establish a correspondence between the main interface objects identified in the analysis model and the interface agents of the GUI models. So, a framework is used [LM 96] to implement each agent (see Figure 4 and 5), according to the particular model used.

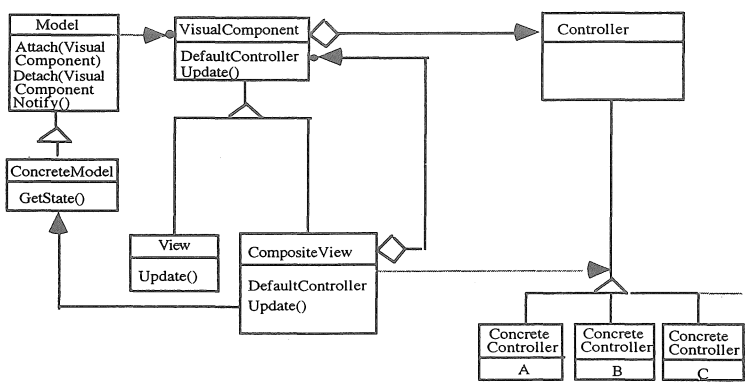


Figure 4. MVC agent framework

In particular, we are interested on the actor window agents and on the use-case windows of this actor, according to the schema defined in Section 3. These are the interface objects in the first level of the interface component relative to an actor. Hence, these interface objects are included in the framework shown in Figure 6.

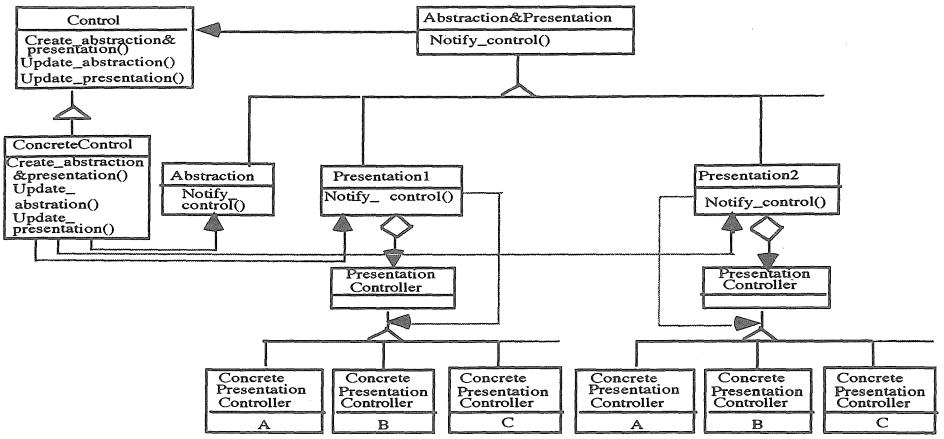


Figure 5. The PAC agent framework

So, this framework is pugged into the Agent framework for window of $Actor_a$, shown in Figure 3. Assuming one of these solutions, ControlClass $Actor_a$ interacts with the GUI agent relative to actor *a*, through the particular ConcreteModel $Actor_a$ window class, if the MVC model is selected, or with ConcreteControl $Actor_a$ window class, if the PAC model is selected.

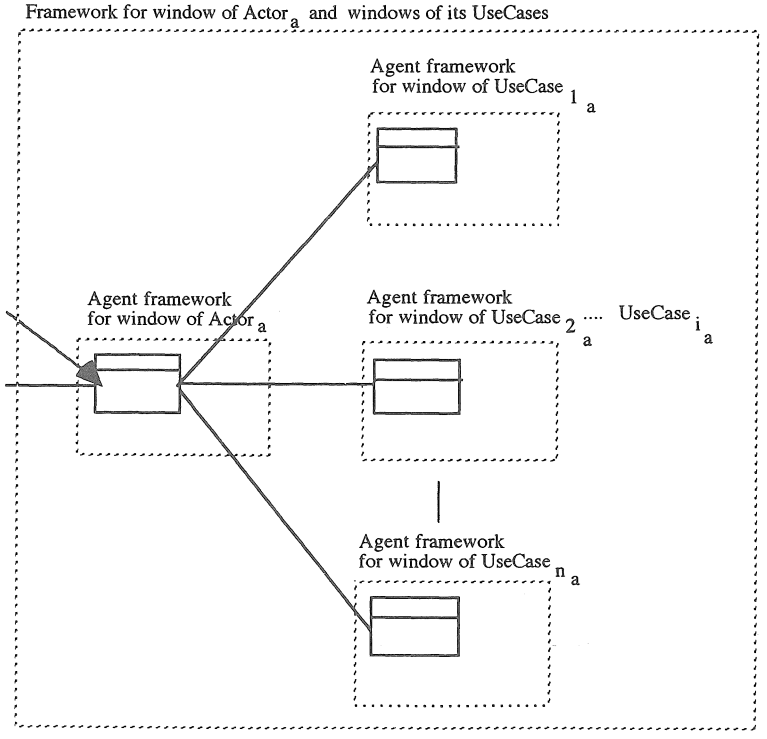


Figure 6. Framework for an actor's window and its use-case windows

4.3 Generic framework for an object-oriented system architecture based on use-cases. In this section a generic framework is defined for expressing the system architecture. We have seen that the frameworks defined in Sections 4.1 (Figure 3) and 4.2 (Figure 6) can "plugged" in, providing the design of the control and interface objects associated with a system's actor. On these basis, considering the human actors and their respective use-cases expressing the GUI functionalities, we can define the genric framework shown in Figure 7. In this framework the SystemTop-LevelControl class is responsible for the activation of the subsytem corresponding to an actor, which requires to interacts with the system, at a given moment. The requirement is captured through the interface, offering the whole system presentation, corresponding to the framewark called Agent framework for SystemView Window.

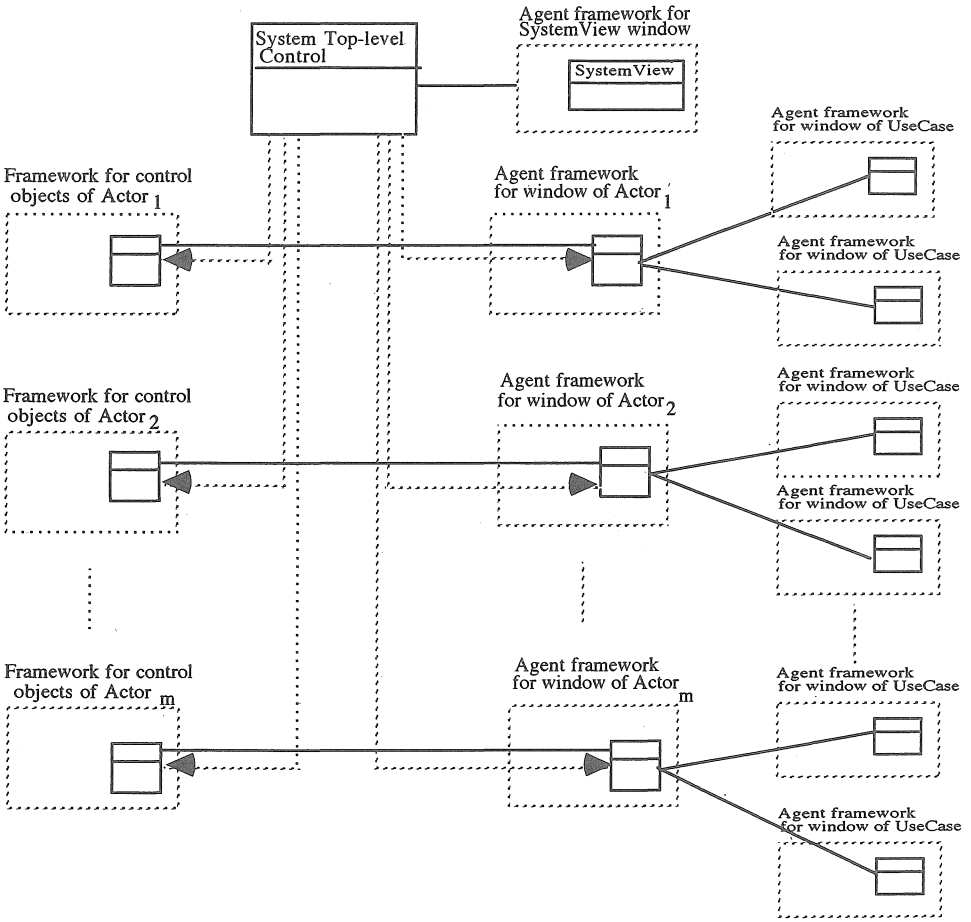


Figure 7. Generic framework for an object-oriented system architecture, according to a use-case based design

5. CONCLUSION

This work has presented a generic framework for a system architecture, according to a use-case based design. This framework can be reused for any system in the design step, for establishing the respective model, according to Jacobson's OOSE method. It also establishes a clear traceability among the control and interface objects identified in the analysis model and the respective classes implementing them. Moreover, the system architecture is completely determined as a function of the identified human actors and their corresponding use-cases. On the other hand, the architecutre obtained through this framework allows a clear differentiation of three layers, organizing the classes implementing the OOSE objects, corresponding respectively, to the problem domain, the control and interface components. This organization enhances a total separation between the problem domain and the interface component, which communicate indirectly trhough the control component. From these considerations, it is evident that a structure easing reuse, maintenace and portability of the system is obtained. Teamwork is also facilitated during the development phase. The system may be seen as a set of sybsystems, one for each actor. The interface agent notion is present in multiagent models such as MVC and PAC and is designed by the respective framework implementing the agent. According to the selected multiagent model, the architecture of the generic framework is adapted to the framework for the particular model. In consequence, we can conclude that the proposed framework provides a grater level of genericity, in the sense that it is adaptable to the specific framework implementing the interface agent.

6. REFERENCES

[Car 95] CARROL J., M., "Scenario-Based Design", Jhon Wiley &Sons Inc., 1995.

[Cic 84] CICCARELLI E. C., "Presentation Based User-Interfaces", Technical Report 794, Artificial Intelligence Laboratory, Massachussets Intelligence Laboratory, August 1984.

[Col 95] COLLINS D., "Designing Object-Oriented User Interfaces", Benjamin/Cummings Publishing Company, Inc., 1995.

[Cou 90] COUTAZ J., "Interfaces homme-ordinateur", Dunod 1990.

[Del 92] DELMAS S. "Tcl/TK User's Manual", University of California, Berkeley, 1992.

[Jac&Al 92] JACOBSON I., CHRISTERSON M, JONSSON P., ÖVERGAARD G., "Object-Oriented Software Engineering, a Use Case Driven Approach", Addison Wesley, 1992.

[F&D 84] FOLEY D.J., VAN DAM, "Fundamentals of Interactive Computer Graphics", Addison Wesley, 1984.

[Gar 93] GARFINKEL S.L., MICHEL K.M. "The NeXSTEP Programming: Step One, Object-Oriented Applications" A Tutorial for developers using the Objective-C Language and the NeXT Interface Builder. Santa Clara, CA:Telos/Springer-Verlag, 1993.

[Gol 84] GOLDBERG A., "Smalltalk-80 - The Interactive Programming Environment", Addison-Wesley 1984.

[HH 89] HARTSON R., HIX, D., "Human-Computer Interface Development Concepts and Systems for its Management", ACM Computing Surveys, Vol.21, No.1, March 1989.

[Lan 86] LANTZ K. A., "On User-Interface Reference Models", ACM SIGCHI, Bulletin, vol. 18, 2 (1986), 36-44.

[KP 88] KRASNER G. E., POPE S.T. "A Cookbook for using the Model-View-Controller user-interface paradigm in Smalltalk-80" Journal of Object-Oriented Programming 1(3), 1988.

[LG 95] LOSAVIO F., GALVEZ C.,"The platform influence on object-oriented development based on the multiagent model", Proceedings of the XXI Conferencia Latinoamericana de Informática, Panel'95, Canela RS, Brasil, August 1995.

[LM 95] LOSAVIO F., MATTEO A. "Use-Case and Multiagent Models for Object-Oriented Design of User-Interfaces" LRI, Orsay, France, Research Report No. 992, September 1995. To appear in Journal of Object Oriented Programming

[LM 96] LOSAVIO F., MATTEO A. "Object-Oriented User-Interfaces Design Based on Agents' Frameworks" Centro ISYS, Caracas, Venezuela, Reporte de Investigación No. 1, RI/01/96, Enero 1996.

[Los 95] LOSAVIO F. "An Object-Oriented Methodology for User-Interface Design Based on the Multiagent Model", Proceedings of the Information Systems Analysis Synthesis, ISAS'95, 89-92, Baden-Baden, Germany, August 1995.

[Los&Al 94a] LOSAVIO F., MATTEO A., ORDAZ O., MEZA O., GONTIER W. "An implementation of the PAC architecture using object-oriented techniques" Proceedings IFIP'94, 13th World Computer Congress 94, Volume 2, 149-155. Hamburgo-Alemania, Agosto 1994, K. Brunnstein and E. Raubold (Editors) Elsevier Science B.V. (North-Holland)

[Los&Al 94b] LOSAVIO F., MATTEO A., ORDAZ O., MEZA O., GONTIER W. "Object-oriented approach and PAC model: design of the GReAt (Graph Researcher Assistant) environment" Proceedings XX Conferencia Latinoamericana de Informática, Panel'94, 433-443, Monterrey, Estado de México, Septiembre 1994

[LVC 89] LINTON M.A., VLISSIDES J.M., CLADER V., "Composing User-interfaces with Interviews", IEEE, Computer, 8-22, Feb. 1989.

[Mey 88] MEYER B., "Object-Oriented Software Construction", Prentice Hall 1988.

[Ste 87] STERN H. L. , "Comparison of Window Systems" BYTE 12, 11, Nov. 1987.

[Str 93] STROUSTRUP B., "The C++ Programming Language", Second Edition, Addison-Wesley Publishing Company, 1993.

Flavio Araujo de Mattos *é Mestre pelo Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ (Universidade Federal do Rio de Janeiro) e Bacharel em Matemática (modalidade informática) pela UFRJ. Trabalha na área de informática desde 1984, tendo ministrado cursos e palestras para empresas. Suas áreas de interesse são: Engenharia de Software, Banco de Dados, Reutilização de Software e Orientação a Objetos. Atualmente, é analista de sistemas do BNDES.*

Cláudia Maria Lima Werner *é Professora Adjunto do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ (Universidade Federal do Rio de Janeiro). Doutora em Eng. de Sistemas e Computação pela COPPE/UFRJ e Bacharel em Matemática (modalidade informática) pela UFRJ. Trabalha na área de informática desde 1983, tendo ministrado diversos cursos de pós-graduação na COPPE e cursos/palestras para empresas. Suas áreas de interesse são: Engenharia de Software, Reutilização de Software, Orientação a Objetos e Ambientes de Desenvolvimento de Software. Já publicou diversos artigos técnicos em congressos nacionais e internacionais.*

Uma Abordagem Orientada a Objetos para Reutilização baseada em Linguagens de Domínio

Flavio Araujo de Mattos
Cláudia Maria Lima Werner

COPPE - Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
Caixa Postal 68511
CEP. 21945-970 - Rio de Janeiro, RJ - Brasil

Resumo

Este artigo apresenta uma abordagem de integração das tecnologias compositiva e gerativa de reutilização, mais especificamente as tecnologias de orientação a objetos e a de sistemas baseados em transformação, obtendo flexibilidade e manutenibilidade das tecnologias compositivas, juntamente com um alto grau de abstração e facilidade de uso das tecnologias gerativas. A abordagem dá ênfase ao papel das linguagens de alto nível orientadas a domínios de aplicação.

Abstract

This paper presents an approach for the integration of compositive and generative reuse technologies, more specificaly the object oriented technology and transformation based systems, joining flexibility and maintainability from compositive technologies, and high level abstraction and ease of use from generative technologies. The approach emphasizes the role of application-domain oriented high-level languages in software reuse.

1. Introdução

A reutilização de software, mais que uma nova linha de pesquisa dentro da engenharia de software, é uma necessidade principal dada a crescente demanda por sistemas maiores e mais complexos. Traduzida em termos de limitações de custo, tempo, qualificação e disponibilidade de recursos humanos, esta necessidade transcende as fronteiras da reutilização de código objeto e atinge o processo de desenvolvimento em sua totalidade.

O grande desafio das propostas de amplo emprego da reutilização é abranger a enorme diversidade de formas e tecnologias através das quais o software se manifesta. Uma solução de reutilização a médio e longo prazo deve, ao mesmo tempo, dar apoio às principais linhas tecnológicas hoje adotadas e ser extensível para capturar as novas que virão.

A orientação a objetos, além de oferecer mecanismos específicos úteis ao reutilização (como a herança e o polimorfismo) [Booc94], traz a seu favor o poder de conciliar tecnologias de diversas áreas a que o paradigma já está integrado e frutificando, tais como linguagens de programação, bancos de dados orientados a objetos, métodos de análise e projeto, entre outras. Este caráter interdisciplinar é fundamental para a pesquisa em uma área de tão amplo espectro como a reutilização.

Em [Wern92], propôs-se como continuidade de pesquisa em reutilização a elaboração de uma abordagem híbrida, que combinasse fortemente aspectos gerativos e compositivos. Tal abordagem híbrida permitiria unir o alto nível de abstração e facilidade de uso das tecnologias de geração com a flexibilidade e manutenibilidade das tecnologias de composição. As tecnologias de composição cumpririam, nesta abordagem, o papel de intermediar o refinamento da especificação de alto nível até a construção de um sistema, especificando partes reutilizadas no processo de geração.

Este artigo apresenta uma abordagem de integração das tecnologias de orientação a objetos e a de sistemas baseados em transformação [Matt95]. Esta abordagem dá ênfase ao papel das linguagens de alto nível orientadas a domínios de aplicação, provendo uma nova leitura do processo de desenvolvimento de software como sendo uma evolução de conceitos abstratos dirigida por linguagem, em analogia aos processos intelectuais em que a linguagem exerce o papel de ordenador do pensamento.

O artigo está dividido em cinco seções. Na seção 2, introduzimos a questão da reutilização de software, apresentando as tecnologias de reutilização e a importância da linguagem na reutilização. Na seção 3, apresentamos uma abordagem para reutilização baseada em linguagens voltadas para domínios de aplicação. Na seção 4, é descrita a base conceitual sobre a qual esta abordagem foi elaborada. Finalmente, concluímos o artigo indicando trabalhos em andamento.

2. Reutilização

O desenvolvimento de software é muito mais que a elaboração de programas. Antes e depois desta etapa há um grande esforço intelectual despendido em atividades organizadoras distribuídas ao longo do ciclo de vida de um software. Durante estas atividades, o engenheiro de software constrói novas abstrações e trabalha com as de que já dispõe por experiência ou educação. A ausência de reutilização é percebida quando é observado que, enquanto a experiência prévia de um indivíduo é integralmente reaplicada em problemas similares, os frutos desta experiência não o são.

Diante da crise de software, a perspectiva de aumento de produtividade pela capacidade de aproveitar produtos de esforços anteriores delineia um horizonte onde a produção de software assuma feições de produção industrial, com benefícios de qualidade e produtividade. Enquanto o desenvolvimento se dá em pequena escala, o âmbito do problema é meramente tecnológico. Quando a experiência a

reutilizar transcende a de uma pessoa ou equipe fixa e pretendemos exercer a reutilização de modo intensivo e por um período arbitrariamente grande de tempo, temos caracterizado a reutilização em larga escala. Neste caso, somente a aplicação de técnicas e disciplinas específicas permite avaliar com segurança se algum trabalho pode ser reaproveitado com esforço menor ao do desenvolvimento a partir do zero. A viabilidade da solução de reutilização em larga escala envolve mudanças no modo atualmente empregado para a construção de software. A busca das soluções que viabilizem a reutilização é o alvo da pesquisa denominada Engenharia de Software Reutilizável (ESR) [Free87].

Neste trabalho adotamos uma visão ampla do que é software, incluindo em um sistema o conhecimento usado para desenvolvê-lo. Nesta visão, não há desenvolvimento sem reutilização, mas há a distinção de reutilização informal, que é totalmente subjetivo em qualidade e produtividade, e da reutilização formal, que repousa sobre instrumentos de apoio e cuja aplicação é desejável em larga escala.

A reutilização formal vem sendo amplamente descrita como uma área onde o sucesso é a exceção e não a regra [Prie93]. A percepção de que a reutilização formal não é largamente empregada serve de estímulo a novas experiências e à produção de tecnologia para transformar a reutilização informal em reutilização formal.

A reutilização formal caracteriza-se, portanto, pela intenção de reutilizar sistematicamente. Esta intenção pode ser traduzida no cuidado a cada passo da evolução de um sistema de software, aproveitando ao máximo as experiências prévias similares. Para que o aproveitamento seja máximo, a experiência prévia deve estar de algum modo codificada em diversos níveis de abstração para reutilização futura.

2.1. Tecnologias de Reutilização

Tradicionalmente, as tecnologias para reutilização são divididas segundo dois princípios fundamentais de reutilização: composição e geração [Bigg87]. O princípio da composição é caracterizado pela criação de software através da reutilização de componentes atômicos e idealmente não modificáveis no curso de sua reutilização. O princípio da geração baseia-se na reutilização de seqüências de instruções e seqüências de transformações embutidas em um gerador, sendo possível que o produto gerado seja bem diferente das seqüências que lhe deram origem.

São exemplos de reutilização compositiva as tecnologias baseadas em bibliotecas de sub-rotinas e orientação a objetos. Sob a reutilização gerativa estão as linguagens de altíssimo nível, os geradores de aplicação e os sistemas baseados em transformação.

Estas duas categorias, entretanto, não são estanques, sendo a fronteira entre o compositivo e o gerativo cada vez mais tênue e difusa, na medida em que são elaboradas soluções que mesclam características de ambas as linhas.

2.2. A importância da linguagem na reutilização

Os produtos materiais do desenvolvimento de software (como programas, documentos e diagramas) são reflexos da atividade intelectual que os produziu. Pensar que a reutilização pode ater-se a este material produzido é incorrer no erro de perder sua essência intelectual. É dissociar forma e conteúdo. A dinâmica da reutilização em qualquer de suas formas sempre recai em, a partir da representação, apreender os conceitos abstratos que ela representa e deles derivar uma nova forma que nos interessa. Cabe à tecnologia facilitar esta transformação, de modo que o que foi matéria prima para a mente, também, o seja para a representação.

Uma das mais importantes linhas de pesquisa para a reutilização é o desenvolvimento de linguagens de muito alto nível que propiciem uma forma de expressão intelectualmente natural dentro de um domínio específico do conhecimento. Esta abordagem possui forte respaldo na ciência da cognição, onde pensadores como Vigotsky apresentam o pensamento como uma processo fortemente guiado pela linguagem. "A formação de conceitos (...) é dirigida pelo uso de palavras como meio de centrar ativamente a atenção, abstrair determinados traços, sintetizá-los e simbolizá-los por meio de um signo" [Vigo87]. Estas linguagens de altíssimo nível expressam o conhecimento como um padrão de atribuição de significados a suas sentenças.

Dentro desta linha uma das mais importantes propostas foi a estratégia DRACO [Neig89], que lançou as bases para a análise de domínio[1] e provê um suporte genérico à construção de linguagens para domínios de aplicação. A abordagem DRACO prevê que cada domínio da aplicação é representado por uma linguagem específica, que serve como organizador de objetos e operações modelados a partir do domínio. A linguagem de domínio passa a ser exercida no desenvolvimento de sistemas para aquela área, oferecendo o que Neighbors caracteriza como *reutilização de análise*. Trabalhos de aperfeiçoamento desta abordagem vem sendo realizados trazendo novas perspectivas [Leit92][Leit94].

3. A Abordagem Gremlin

Nesta seção, apresentamos a abordagem Gremlin, cujo nome representa nosso objetivo principal que é o de gerar reutilização por mecanismos baseados em linguagens.

Gremlin é uma abordagem que se alinha com a estratégia DRACO, no sentido de enfatizar o papel da linguagem de alto nível orientada ao domínio de aplicação na reutilização de software. Gremlin, à semelhança do DRACO, oferece níveis de estratificação de conceitos que mapeiam domínios de aplicação, para formar espaços de trabalho dentro dos quais a solução abstrata para um problema é especificada. Tanto a construção destes espaços de trabalho, como os mecanismos de tradução das especificações de alto nível em especificações progressivamente mais próximas da máquina foram desenvolvidos para explorar as vantagens do paradigma da orientação a objetos (O.O.)

Para isto, foi desenvolvido um modelo de objetos particular, que acrescenta às características tradicionais de O.O. um suporte mais rico às transformações. Este modelo estendido foi integrado à abordagem de reutilização orientada a domínios de aplicação, servindo como ferramental básico da caracterização destes domínios.

A reutilização na abordagem Gremlin se dá em dois níveis. No primeiro nível, componentes de software orientados a objetos são reutilizados na representação de novos domínios de aplicação e na evolução deste domínio no tempo. A orientação a objetos é empregada para representar conceitos de análise e alternativas de projeto ("design") disponíveis para a solução de problemas neste domínio de aplicação. O modelo estendido é empregado, ainda, na caracterização das linguagens de alto nível especializadas no domínio. O segundo nível de reutilização é obtido pelo uso das linguagens de domínio para a construção de soluções de problemas.

---

[1] A análise de domínio é a atividade que organiza o conhecimento sobre a construção de software em uma determinada área [Aran91].

3.1. O Modelo Orientado a Objetos Estendido

Um modelo de objetos é a sistematização dos princípios da orientação a objetos para fins específicos, geralmente associado a uma linguagem de programação, banco de dados ou ferramenta de software que dá suporte operacional a ele. Cada modelo de objetos possui sua definição particular para os conceitos da orientação a objetos. Estas divergências tornam as linguagens orientadas a objetos diferenciadas, de um modo similar ao que acontece com as linguagens de programação convencionais. As diferenças entre estes modelos privilegiam a construção de representações de uma determinada classe de problemas em detrimento a outras. O que varia é a naturalidade com que as soluções de uma área são construídas em cada classe de problemas.

Os modelos orientados a objetos convencionais não dão suporte específico à transformações de programas ou de especificações. Um ambiente orientado a objetos que promove a reutilização baseada em geração convive, por isto, com dois universos conceitualmente distintos. Em um deles está a orientação a objetos, cujas facilidades gerais são usadas para a implementação da vuncionalidade do ambiente e da representação interna das especificações. Em outro está o modelo de transformações, construído como uma camada conceitual grandemente independente da orientação a objetos subjacente, em que a reutilização é exercido.

Um modelo de objetos estendido para dar suporte específico a transformações promove a integração destes dois universos, permitindo a elaboração de ambientes de reutilização que explorem o melhor de cada abordagem. Os requisitos identificados para tal modelo são:

1. Mecanismo geral de transformação - O modelo estendido deve ser dotado de um mecanismo genérico de transformação, bem como de instrumentos para especializar estes mecanismos. Deste modo, mecanismos específicos de reutilização gerativa podem ser criados.
2. Homogeneidade conceitual - As extensões do modelo devem estar homogeneamente integradas às demais características da orientação a objetos, evitando a distinção conceitual do que seja ou não gerativo.
3. Fidelidade ao paradigma da orientação a objetos - As características de herança, encapsulação, cooperação através de mensagens e polimorfismo devem ser preservadas. Isto garante que não sejam perdidos o poder de expressão e a generalidade da orientação a objetos.

O modelo de objetos estendido foi criado a partir de uma revisão do mecanismo de troca de mensagens entre objetos. De um modo geral, os modelos de objetos são muito similares no modo de implementação da troca de mensagens, segundo o padrão definido na figura 1. Inicialmente, o emissor codifica a mensagem como um seletor, usualmente um nome ou um número que a identifica. Opcionalmente, objetos complementares podem ser agregados a ela na forma de argumentos. Quando o receptor recebe a mensagem, ele usa uma função de decodificação para determinar o código que será ativado. Ao fim da execução, um valor é retornado ao emissor como resposta.

Buscando uma analogia com os mecanismos encontrados na comunicação humana, observamos que a estrutura das mensagens são de organização extremamente complexa. Dentro desta complexidade, são estabelecidos os relacionamentos entre os significantes, de modo que o significado final da comunicação possa ser construído pelo receptor da mensagem. Apenas em poucos casos, a interpretação de uma mensagem é imediata como na orientação a objetos. Na maioria das vezes, a compreensão do significado de uma mensagem é obtida por processos intelectuais superiores, intimamente ligados ao processo de pensamento e à linguagem humana [Vigo87].

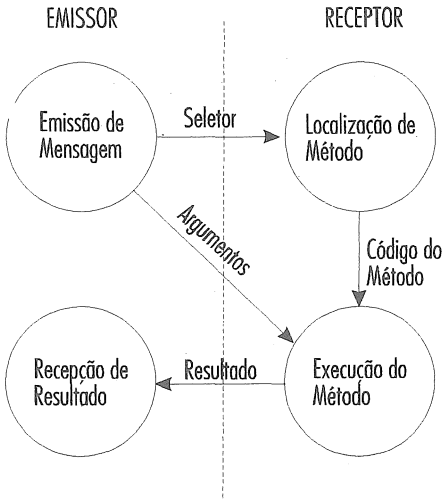


Figura 1 - Mecanismo tradicional de troca de mensagem entre objetos

Portanto, uma das formas para se obter os requisitos anteriormente identificados para um modelo de objetos estendido é aproximarmos o mecanismo de troca de mensagens à comunicação humana. Tal aproximação envolve: a adoção de mensagens de estrutura complexa, um mecanismo mais poderoso de interpretação das mensagens e a encapsulação deste mecanismo em uma linguagem que define a organização das mensagens que um objeto pode receber.

A figura 2 mostra o modelo de troca de mensagens estendido. Neste modelo, as mensagens de um objeto são descritas por uma linguagem livre de contexto, cujo mecanismo de transformação é regido por procedimentos associados à aplicação de produções da linguagem. O paradigma de envio de mensagens permanece muito próximo da orientação a objetos tradicional. A interpretação da mensagem é um método de compilação tradicional, que passa por sucessivas transformações da mensagem original até que um resultado seja completamente definido.

3.2. Domínio de Aplicação no enfoque Gremlin

Para desenvolver software de qualquer tipo usamos, mesmo que informalmente, a especificação de requisitos do que ele fará. A necessária interpretação desta especificação requer sempre informações sobre o contexto em que o software está inserido. Hoje, reconhecemos que este conhecimento sobre o contexto pode ser organizado segundo a área do conhecimento ou da tecnologia a que cada software se presta. Este conhecimento, organizado e sistematizado, dá origem a um grande potencial de reutilização que pode ser exercido de diversos modos. A própria organização do conhecimento consiste em elemento passível de reutilização. O maior ganho, porém, advém da construção de software reutilizável que dê suporte operacional a esta área.

Denominamos *domínio de aplicação* a área do conhecimento que se serve ou se servirá de software, e que pretendemos organizar para a reutilização. Nesta definição, caracterizamos um ponto de vista em que os domínios de aplicação não pré-existem à reutilização. O domínio de aplicação apenas nasce quando há a percepção do seu potencial de reutilização. As técnicas de organização do conhecimento em domínios de aplicação são alvo da pesquisa em *análise de domínios*. Hoje, existem diversas propostas de métodos para análise de domínios [Aran94].

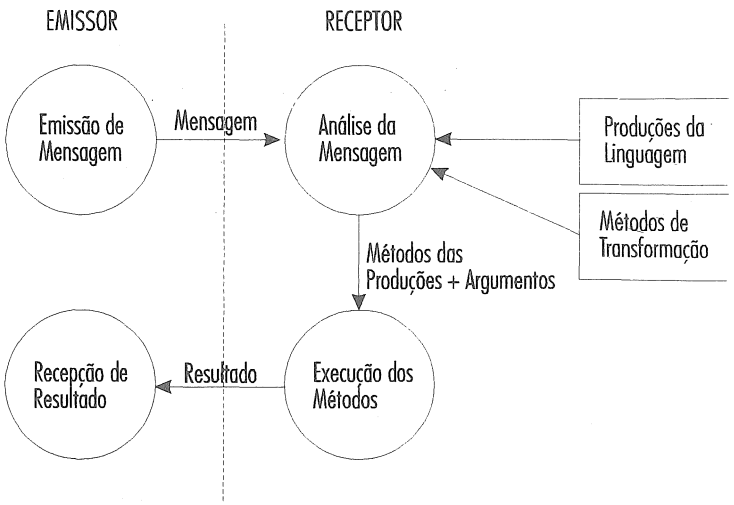


Figura 2 - Extensão do mecanismo de troca de mensagens

Um ambiente para a reutilização em larga escala deve prover uma base conceitual em que os diferentes métodos de análise de domínio possam se encaixar. Na próxima seção, apresentamos a base conceitual utilizada pela abordagem Gremlin para a reutilização em domínios de aplicação específicos.

4. Patamar de Reutilização, o representante Gremlin

A abordagem Gremlin agrupa as informações reutilizáveis em ambientes de trabalho, denominados *patamares de reutilização*. Para cada domínio de aplicação identificado, há um único patamar de reutilização. Um patamar de reutilização concentra todas as informações consideradas relevantes em um domínio de aplicação, incluindo documentação, uma base de conceitos do paradigma e componentes de software que especificam, operacionalmente, a visão atual que se tem deste domínio. O patamar de reutilização cumpre a função de organizar esta diversidade de informação em um ambiente de desenvolvimento coerente.

Um patamar de reutilização pretende ser um ambiente suficiente para o desenvolvimento de soluções de software em um domínio de aplicação específico. Para conquistar este objetivo, é necessário que dê suporte básico a várias atividades dentro do ciclo de vida do desenvolvimento. A fim de atender a diversos ciclos de vida, estabelecemos como requisitos fundamentais de um patamar de reutilização:

- Apoiar o processo de compreensão de conceitos do domínio, bem como dos instrumentos através dos quais são reutilizados;
- Representar modelos de conceitos do domínio de aplicação, incluindo modelos operacionais, que possam ser empregados na construção de software; e
- Representar linguagens de desenvolvimento orientadas ao domínio de aplicação.

As tomadas de decisão envolvidas em um processo de reutilização exigem que o engenheiro de software conheça os conceitos do domínio de aplicação envolvidos no processo. Por isto, o suporte à compreensão é fundamental em um ambiente de reutilização. A abordagem Gremlin apóia o estudo dos conceitos de um domínio de aplicação através da capacidade de registrar documentos em

formato livre em uma rede de hipertexto. Navegando por esta rede, o usuário adquire o conhecimento sobre o domínio de aplicação, capacitando-se para o processo de reutilização.

Dentre as possíveis representações de um domínio de aplicação, a construção de modelos operacionais é aquela que oferece resultados à reutilização em prazos mais curtos. Um modelo operacional registra aspectos estruturais e comportamentais do domínio de aplicação, de modo que pode ser empregado em protótipos e em avaliações de qualidade. A abordagem Gremlin usa a orientação a objetos para representar modelos operacionais. Deste modo, um modelo operacional representa arquiteturas do domínio de aplicação e padrões de interação entre objetos do domínio, similarmente às abordagens que empregam a tecnologia de *frameworks*[2] [Cima94]. A maturidade de um modelo operacional orientado a objetos é obtida quando este é formado por componentes reutilizáveis, que podem ser empregados na construção de soluções de software dentro do domínio de aplicação.

O papel das linguagens de alto nível é o de oferecer um mecanismo para descrever soluções de software, abstraindo-se dos detalhes irrelevantes à concepção da solução. Estes detalhes são supridos durante o processo de tradução da linguagem, seguindo padrões de complexidade variável. Retirando estes detalhes da atenção do engenheiro de software, a linguagem de alto nível dirige sua atenção para as características relevantes do domínio de aplicação. Este processo de tradução poupa trabalho ao engenheiro de software, aumentando sua produtividade e reduzindo as áreas onde pode haver a incidência de erro humano. Ao direcionar a atenção, também, influi positivamente a capacidade de criar soluções [Vigo87].

Para atender aos requisitos enumerados, o patamar de reutilização é composto por quatro subsistemas (figura 3):

- **Subsistema de documentação** - responsável pela armazenagem, organização, classificação e acesso a documentos em formato livre;
- **Subsistema de modelagem** - responsável pela construção e implementação de modelos de software, destinados a representar conceitos de domínios de aplicação como arquiteturas de software reutilizável;
- **Subsistema lingüístico** - responsável por construir linguagens de alto nível orientadas a domínios, para a especificação de soluções de software, e por traduzir estas especificações em termos de reutilização de arquiteturas já definidas; e
- **Subsistema Compositivo** - responsável por classificar e organizar os diversos tipos de informação (documentos, modelos e linguagens) de modo útil à reutilização.

4.1. Subsistema de Documentação

O subsistema de documentação provê mecanismos para registrar e tornar disponíveis informações pertinentes ao exercício da reutilização dentro de um patamar de reutilização. A unidade de armazenamento de informação é chamada *documento*. Dentro de um patamar de reutilização, um documento é qualquer objeto informativo, que pode ser examinado ou alterado através de uma interface adequada.

---

[2] Frameworks consistem de classes que foram especialmente projetas para serem refinadas e usadas em grupo [Wirf90].

Os documentos integram-se aos demais objetos dentro do patamar de reutilização em uma rede. O engenheiro de software interessado em reutilização navega por ela como em um hipertexto, de modo a atingir os documentos ou outros objetos que deseje.

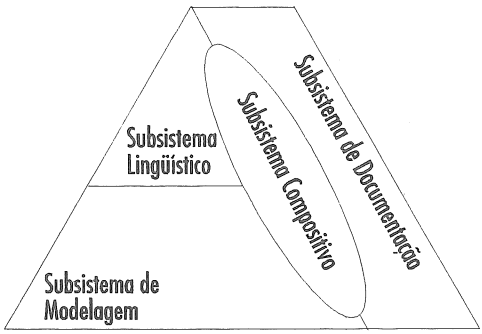


Figura 3 - Relacionamento entre os subsistemas de um patamar de reutilização

4.2. Subsistema de Modelagem

Uma vez que os modelos de objetos são a forma de representação mais importante para a reutilização dentro de um patamar de reutilização, sua construção deve avançar em conjunto com o processo de análise de domínio. Por isto, o subsistema de modelagem precisa dar suporte a ciclos iterativos de desenvolvimento de modelos. Nestes ciclos, a construção de um modelo surge pelo refinamento sucessivo a partir de um modelo abstrato e incompleto. A cada refinamento, o modelo de objetos incorpora mais informações colhidas da análise do domínio de aplicação. É inevitável que, neste processo de refinamento, algumas decisões de representação sejam tomadas. Para reduzir o impacto destas decisões na generalidade da representação do domínio em consideração, é importante avaliar as alternativas disponíveis.

O subsistema de modelagem representa cada modelo de um patamar de reutilização como uma entidade distinta, composta de classes e objetos inter-relacionados. O processo de refinamento é exercido sobre estas entidades e pode envolver a construção de versões alternativas para a avaliação das decisões de representação consideradas.

Um modelo é dito *prototipável* quando a definição das classes é suficientemente rica para a construção de protótipos de soluções. O papel dos protótipos é levantar aspectos específicos do desenvolvimento de aplicações no domínio, baseado nos conceitos representados pelo modelo do domínio de aplicação.

Um modelo é dito *maduro* quando suas classes constituem componentes de software cuja qualidade seja julgada adequada para a reutilização dentro de um domínio de aplicação. Cada patamar de reutilização disponível para o desenvolvimento de software está associado a um único modelo de objetos, necessariamente maduro. Este modelo representa todos os aspectos do domínio de aplicação disponíveis para reutilização.

Para cada linguagem de um patamar de reutilização, há pelo menos uma gramática que reconheça sua sintaxe. Quando duas ou mais gramáticas reconhecem a mesma linguagem, elas são ditas *gramáticas polimórficas* e representam alternativas de tradução para uma mesma linguagem. O nome "gramática polimórfica" vem do modelo de objetos subjacente, que trata a tradução de sentenças como envio de mensagens. No sentido rigorosamente O.O., dois objetos que atendem de modo diferente a mesma mensagem são ditos polimórficos. Uma exemplo trivial de gramáticas polimórficas são gramáticas que traduzem uma especificação de alto nível em diferentes linguagens de baixo nível.

Componentes gramaticais ou gramáticas abstratas são fragmentos de linguagens que são usados para construir novas gramáticas, ou para criar novas versões de gramáticas já existentes, de modo a alterar sua sintaxe ou sua semântica de tradução. As gramáticas, abstratas ou não, são objetos da mesma natureza, com a diferença que a definição das gramáticas abstratas é incompleta, impedindo que ajam como elementos transformadores de sentenças.

4.4. Subsistema Compositivo

O subsistema compositivo tem por finalidade prover o instrumental básico para a obtenção de reutilização compositiva na construção das estruturas complexas que constituem um patamar de reutilização, proporcionando a reutilização da experiência prévia na estruturação de patamares de reutilização. Este subsistema é responsável por organizar grandes conjuntos de componentes de software, sejam eles documentos, classes ou gramáticas. Além disto, oferece mecanismos de busca e seleção de componentes, de modo que estes são tornados disponíveis ao usuário. O exercício específico da integração destes componentes em uma organização é realizado dentro de um dos demais subsistemas.

O processo de seleção de componentes é baseado em um esquema de classificação por facetas [Prie91]. Esta classificação auxilia a exploração seletiva de conjuntos de componentes, de modo a identificar aqueles que contribuem ao processo de reutilização.

5. Conclusões

Neste artigo apresentamos uma abordagem para a reutilização de software orientada a domínios de aplicação. A abordagem baseia-se no emprego de técnicas compositivas, oriundas da orientação a objetos, na construção de modelos genéricos sobre domínios de aplicação, cuja manipulação é regida por linguagens especializadas construídas para estes domínios.

Esta abordagem compreende a especificação de um modelo de objetos que dá suporte nativo a mecanismos transformacionais e caracteriza um modelo de extensão do modelo baseado no enriquecimento do mecanismo de troca de mensagens. Sobre tal modelo elaborou-se uma proposta de organização de domínios e de linguagens especializadas que aproveita a reutilização compositiva em sua construção e a reutilização gerativa na criação de sistemas aplicativos.

Para dar suporte operacional à abordagem descrita neste artigo, foi desenvolvido um protótipo de ferramenta, escrito na linguagem C++, em microcomputadores 486 [Matt95]. Neste momento, estão sendo realizados experimentos que possibilitarão a avaliação empírica da abordagem apresentada, dentro de domínios de aplicações específicos. Esta avaliação contempla o esforço de aprendizado das técnicas apresentadas, os benefícios e os problemas eventualmente encontrados.

Os componentes de software reusável encontrados em um modelo maduro de um domínio de aplicação são chamados *componentes semânticos*. Estes componentes dividem-se em duas grandes categorias. A mais importante engloba os *componentes de domínio*, que descrevem aspectos relevantes do domínio. A segunda categoria de componentes semânticos refere-se aos *componentes auxiliares*, que se prestam a tarefas secundárias, mais relevantes às implementações de um modelo, à verificações de consistência, etc.

Cada patamar de reutilização possui um conjunto de componentes peculiar, chamados de *componentes fundamentais*. O que caracteriza um componente fundamental não é o seu caráter semântico ou auxiliar, e sim o fato de não possuir definição dentro do patamar de reutilização em que está inserido. A definição de um componente fundamental está em outro patamar de reutilização (e, portanto, seu significado reside em um domínio de aplicação diverso). Estes componentes servem de ponte entre este domínio e domínios de nível mais baixo, estabelecendo uma ordem parcial entre os domínios semelhante à criada na abordagem DRACO[3].

4.3. Subsistema Lingüístico

O subsistema lingüístico é responsável por oferecer o potencial de reutilização do modelo de objetos do domínio de aplicação através de linguagens de desenvolvimento especializadas. O analista de sistemas comum vê um patamar de reutilização como um ambiente de desenvolvimento, dotado de ferramentas de apoio à construção de sistemas. Através deste ambiente, ele tem acesso a uma ou mais linguagens de desenvolvimento voltadas àquele domínio de aplicação. Estas linguagens e ferramentas, todavia, compartilham a mesma base conceitual que representa o domínio de aplicação, com a maior fidelidade possível.

A proposta da multiplicidade de linguagens de desenvolvimento em um domínio vem de diversos fatores. Em primeiro lugar, podemos constatar empiricamente de que os domínios de aplicação já bem caracterizados podem possuir mais de uma linguagem de desenvolvimento. Em segundo lugar, sabemos que uma linguagem de desenvolvimento é um produto tecnológico e, como tal, sujeito a mudanças e evolução. A base conceitual de um domínio de aplicação, ao contrário, tende a assumir um caráter mais estável, de modo que é necessário estabelecer um certo grau de liberdade entre domínio e linguagem de desenvolvimento. Gremlin oferece esta independência relativa, abrindo a possibilidade de que coexistam mais de uma linguagem de desenvolvimento dentro de um mesmo domínio.

A unidade de organização do subsistema lingüístico é a *gramática*. Conceitualmente, uma gramática é uma descrição da sintaxe de uma linguagem do patamar de reutilização, bem como dos mecanismos de transformação de sentenças desta linguagem em objetos do modelo de objetos do domínio.

Operacionalmente, gramáticas são objetos do modelo O.O. estendido, cuja interface reconhece sentenças desta linguagem. Cada gramática tem a capacidade de sintetizar objetos que representam a sentença da gramática recebida, através de transformações encapsuladas em sua implementação.

---

[3] A hierarquia de domínios do DRACO caracteriza um grafo orientado acíclico, onde o refinamento é orientado de domínios de níveis mais alto de abstração (i.e. *domínios de aplicação e domínios de modelagem)* para os de nível inferior (i.e. *domínios executáveis)* [Leit94].

Referências Bibliográficas

[Aran91] Guillermo Arango; Rubén Prieto-Díaz - *"Domain Analysis Concepts and Research Directions"* em Domain Analysis and Software Systems Modeling, (ed.) Rubén Prieto-Díaz & Guillermo Arango - IEEE Computer Society Press, 1991.

[Aran94] Guillermo Arango - *"Domain analysis methods"* em Software Reusability, (ed.) Wilhelm Schäfer, Rubén Prieto-Díaz, Masao Matssumoto - Ellis Horwood, 1994.

[Bigg87] Ted J. Biggerstaff e Charles Richter - *"Reusability Framework, Assessment, and Directions"*, IEEE Software, Vol 4. #2, março de 1987.

[Booc94] Grady Booch - *"Object Oriented Analysis and Design with Applications"*, The Benjamin/Cummings Publishing Company, Inc.,1994.

[Cima94] Alberto M. De Cima, Cláudia M. L. Werner, Alessandro A. C. Cerqueira - *"The Design of Object-oriented Software with Domain Architeture Reuse"*, Third International Conference on Software Reuse, Rio de Janeiro, 1994.

[Free87] Peter Freeman - *"A Perspective on Reusability"* em Tutorial: Software Reusability, (ed.) Peter Freeman, Computer society Press of the IEEE, 1987.

[Leit92] Júlio C. P. Leite, Antonio Prado, Marcelo Sant'Anna - *DRACO-PUC, experiências e resultados de re-engenharia de software*, VI Simpósico Brasileiro de Engenharia de Software, novembro 1992.

[Leit94] Júlio C.P.Leite, Marcelo Sant'Anna, F.G. de Freitas - *"Draco-PUC: A Techonology Assembly for Domain Oriented Software development"*, Third International Conference on Software Reuse, Rio de Janeiro, 1994.

[Matt95] Flavio Araujo de Mattos - *Uma Ferramenta de Reuso por Interconexão de Linguagens,* COPPE/UFRJ, abril de 1995.

[Neig89] James M. Neighbors, *"Draco - A Method For Engineering Reusable Software systems"*, Software Reusability, Volume I, Concepts and Models, (ed.) Ted J. Biggerstaff e Alan J.Perlis, ACM Press, 1989.

[Prie91] Rubén Prieto-Diaz - *"Implementing Faceted Classification for Software Reuse"*, Communications of the ACM, maio 1991.

[Prie93] Rubén Prieto-Diaz - *"Status Report: Software Reusability"*, IEEE Software, maio 1993.

[Vigo87] L. S. Vigotsky - *Pensamento e Linguagem* - Livraria Martins Fontes Editora Ltda, 1987.

[Wern92] Cláudia M. L. Werner - *Reutilização de Software no Desenvolvimento de Software Científico* - Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, março 1992.

[Wirf90] R.J. Wirfs-Brock, R.E. Johnson - "*Surveying Current Research in Object-Oriented Design*", Communications of the ACM, setembro 1990.

A METHOD FOR OBJECT-ORIENTED USER-INTERFACE DESIGN BASED ON USE-CASES AND MULTIAGENT MODELS.*

J. Antoneti, F. Losavio, A. Matteo
Centro de Ingenieria de Software y Sistemas ISYS
Facultad de Ciencias, Universidad Central de Venezuela,
Apdo. 47567, Los Chaguaramos 1041-A,
Caracas, Venezuela
flosavio @conicit.ve / flosavio@anubis.ciens.ucv.ve
amatteo@conicit.ve / amatteo@anubis.ciens.ucv.ve

Abstract

The goal of this work is to present an object-oriented method for user interface design based on the use-case approach and multiagent models. The use-case approach is useful for the analysis step of object-oriented software development enhancing an early separation between entity control and interface objects, and multiagent models are widely employed to describe the arquitecture of interactive systems. Moreover, the method enhances adaptability of the resulting systems.

Keywords: object-oriented analysis, object-oriented design, user-interface design, PAC model, framework, use-case, object-oriented methods, graphical user interface.

1. INTRODUCTION.

The use-case approach is widley used to define the requirements from the user´s needs. Furthermore, in recent years, the graphical user interface (GUI) has been considered as fundamental part of the application in order to support human activities. Nevertheless, not much can be said about the design of the graphical user interface component, for which traditional development techniques are applied, now enriched with a wide use of toolkit libraries [HH 89]. The main goal of this paper is to propose an object-oriented (OO) method for GUI development based on the use-case approach and multiagent models. In particular, the OOSE[1] method [Jac&Al 92] is used as a front-end for the requirements and analysis phase of our method, which makes also use of GUI frameworks. A framework is defined as a reusable semifinished architecture for various application domains [Pre 95]. It is constituted by a set of classes, that may constitute several design patterns, [Gam&Al 95] conceived for working together and represents a generic subsystem that can be instantiated. The instantiation is acheived developing subclasses from the public abstract classes of the framework, called "hot spots" [Pre 95], which are seen as predefined places where application specific parts are composed. A developer using a framework must know the hot spots for a given problem and know how to adapt them to the application's needs. The MVC (Model-View-Controller) model [Gol 84] for building GUI is considered one of the first known frameworks [KP 88]. Besides this introduction and the conclusions, this paper is divided into four sections. Section 2, gives a brief summary of the OOSE method followed by section 3 with an analogy between OOSE analysis objects

* This research is sopported by the New Technology Program of the BID-CONICIT, the CONICIT MOODE Project and the ISYS Center OOMGRIN Project.
[1] Object Oriented Software Engineering

and GUI interface agents.Section 4 introduces a schema for structuring OOSE analysis objects. Finally, section 5 presents the OOMGRIN[2] method, goal of this work.

2. THE OOSE METHOD.

The OOSE method [Jac&Al 92] covers three phases called *analysis*, *construction* and *testing*. The outputs of the *analysis* step are the requirements model and the analysis model, which specifies and defines the system that is going to be built. During this phase the requirements are defined, and all efforts are concentrated into marking the limits of the system and in defining its functionalities. This phase is composed of three elements: the *use-case model*, the *object model* and the *specific descriptions of the system interfaces*. The use-cases model is the key element in OOSE's method, whereby the functionality of the system to be developed is defined (See Figure 1). The task scripts of interfaces is an essential part of the use-case descriptions, showing also how these are to be presented to the users. The object model is proposed by OOSE for developing a logical vision of the system using objects from the problem domain. The purpose of this model is to developed a robust and extensible structure that can act as the basis for system construction. In this work, we will be mostly interested in the analysis step. The *design* model is the refinement and formalization of the analysis model, and takes into account the implementation details. Its goal is to refine the analysis model and to adapt it to the implementation environment, so that its results can be easily passed to the code. The *implementation* model consists of the source code. The information space of this model corresponds to the computer language that is used.

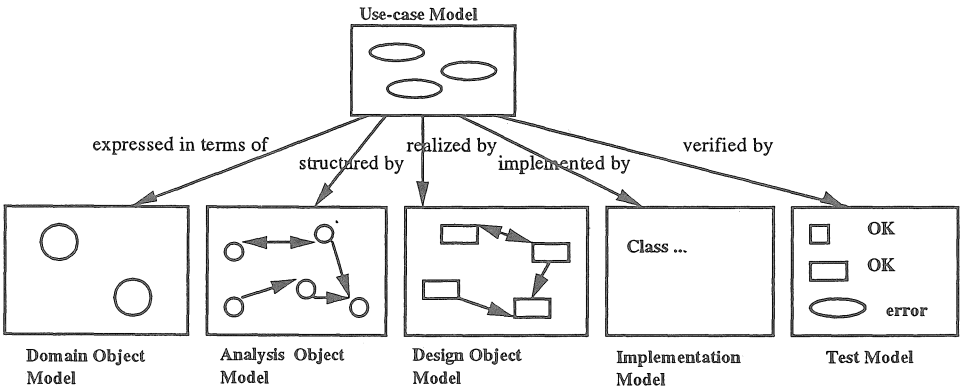


Figure 1. The use-case model encompassing the OOSE system life cycle

The use-case approach expresses the requirements in terms of actors, representing the people, systems and devices that will exchange information with the application. Use-cases represent events en terms of "what happens when" scenarios for each actor. Use-cases provide a common ground for developers and their

[2] Object Oriented Methodology for multiagents model based design of GRaphical INterfaces.

customers to achieve a shared vision of the application, early in the development process, when inconsistencies are easy and inexpensive to correct. The use-cases identified during requirements analysis continue through the whole OOSE life cycle, driving each subsequent phase (see Figure 1). In the Analysis Model, three types of objects for each use-case are identified: *interface*, *entity* and *control objects*. Interface objects (for example graphical user-interface widgets or complex windows), represent objetcs that depend on and change with the application's interface environment. Entity objects represent persistent data of the application, expressing its semantic and control objects contain rules not directly dependent on data or interface. For our approach, we will be interested on the control and interface objects.

3. ANALOGY BETWEEN OOSE ANALYSIS OBJECTS AND AGENTS FOR GRAPHICAL USER-INTERFACE DEVELOPMENT.

In order to construct an interactive system, a textual or graphical description of the user-interface must be provided. These specifications or task scripts express the user requirements for the user-interface of the application to be constructed. The architecture of an interactive system is expressed here by an agent based model [Cic 84], [Gol 84], [LVC 89] (an *agent* is seen as a processor, reacting at external events and generating events). In particular, the PAC (Presentation-Abstraction-Control) model [Cou 90], is constituted by agents, with three perspectives abstraction, control and presentation. PAC maintains the paradigm of the separation of the abstraction or semantic aspects of a problem from the presentation or visual aspects , differring from the well known MVC (Model-View-Controller) model [Gol 84], in the sense that the PAC presentation encapsulates the view-controller MVC pair for capturing and treating external events, and the control notion is introduced for communicating the abstraction with the presentation perspectives. Moreover, communication among the agents (and their perspectives) is only allowed through their respective controls and a hierarchy of agents is established, considering an agent constituted by subagents (as for example a window with menu and palette areas). According to the PAC model, the system architecture is constituted by a top level agent (representing the complete application) situated in the so called *Application level* and the hierachy of GUI agents, in the so called *Interface level.* We could think at first of a semantic analogy between all the objects defined in the OOSE analysis model and the different perspectives of the PAC agents. Nevertheless, since the PAC model is used for building GUI of interactive systems, it can clearly be established that a PAC agent corresponds to an OOSE interface object. Moreover, the three perspectives of a PAC agent correspond to a decomposition of an OOSE interface object: its image is the presentation, the image representation corresponds to the abstraction and the sequence of actions is the control. This correspondence can be established for every PAC agent, excepting the (root) agent on the highest level of the PAC hierarchy, representing the whole application that is being modelled. The abstraction, presentation and control of this agent have a different meaning. The abstraction of the root agent corresponds to all the objects of the application domain (OOSE entity objects). The control perspective of the root agent manages the set of all the system functionalities, particularly those of the user-interface (OOSE control objects). Finally, the presentation perspective of the root agent is in general an icon or window, representing the whole application, from where its functionalities may be accessed. These are represented by icons or menus windows, which are modelled by the agents of the lower levels, corresponding to the OOSE interface objects. The OOSE analysis model is used to derive the PAC model. This new model (to be discussed in Section 5) is called *Use-case-PAC*

Model and represents an architecture of the system's user-interface, where the use-case approach is included. Therefore, the model obtained reflects the involved actors interacting with the system through a GUI [LM 95].

4. A SCHEMA FOR STRUCTURING OOSE ANALYSIS OBJECTS.

A schema for organizing analysis objects in OOSE is presented [LM 95], showing the control and interface objects which have to be defined, considering the different use-cases related with a certain actor and establishing communications among these objects. The balanced control strategy, where the control object is an intermediary between the entity and interface objects, is taken into account in order to derive the schema, which is presented in Figure 2. The lines between the objects represent static links. For the definition of this schema, the following strategies are considered:

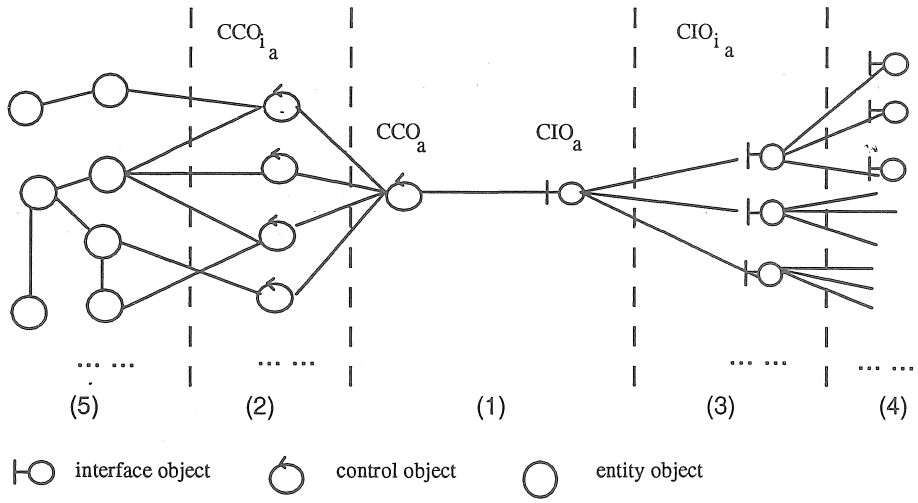
i. Define a central interface object, denoted by CIO_a, associated to the actor, denoted by *a*, corresponding to a window from which an actor can access the corresponding functionality (Level 1 in the schema). Define central interface objects CIO_{i_a}, 1_i_n, where *n* is the number of use-cases associated to *a*, each one corresponding to a window, from which one or more functionality or use-case can be executed (Level 3 in the schema). Then we have at most n CIO_{i_a} .

ii. Define a *central control object*, denoted by CCO_a(Level 1 in the schema). For each functionality or use-case, define a control object, denoted by CCO_{i_a}, 1_i_n, where *n* is the number of use-cases associated to a (Level 2 in the schema). The CCO_a expresses the control requirements of the actor. It manages the distribution of controls among the actor's functionalities, on the basis of the balanced control strategy, allowing also a clear separation between the abstraction and the user-interface aspects. The control object CCO_{i_a} associated to an actor facilitates the possible modifications relative to each functionaliy.

iii. Create independent or completely separated entity objects (Level 5 in the schema) from interface objects. This separation means that they do not "know" each other, in the sense that no links (static or dynamic) are established among them. Notice that the balanced control strategy (see 3.2) is also followed in this case.

iv. Locate toolkit widgets or widgets composition, to construct the windows corresponding to the CIO_{i_a}, 1_i_n (Level 4 in the schema). These objects are non central interface objects.

The above strategies clearly favor the separation between entity and interface objects, easing the system's decomposition into interface and problem domain components. Notice that the OOSE approach does not treat explicitly this separation, allowing links between entity and interface objects [Jac&Al 92]. As a consequence of the proposed schema, easy location of extensions or modifications, and the programmers' teamwork during the implementation phase are facilitated [Los&Al 94a], [Los&Al 94b].



(1) Central control and interface objects associated to the actor (CCO_a, CIO_a respectively)
(2) Control objects, one for each use-case associated to the actor (CCO_{i_a}, 1_i_n)
(3) Central interface objects, associated to the different use-cases of the actor. Each interface object in this level corresponds to a window, allowing the access to one or more functionalities (use-cases) associated to the actor (CIO_{i_a}, 1_i_n)
(4) Interface objects that are not central, such as widgets or objects composed from widgets
(5) Entity objects implied in the solution of the different functionalities (use-cases) associated to the actor

Figure 2. Schema of objects definition with respect to a given "human" actor.

5. AN OO METHOD FOR DEVELOPING INTERACTIVE SYSTEMS.

In [Los&Al 94a], [Los&Al 94b], [LG 95] and [Los 95] some guidelines have been proposed for developing interactive applications, whose architecture is based on the PAC model. They did not impose any particular design method; however, the use of the use-case approach is suggested. Then, in [LM 95] the schema presented in Section 4, has been developed to structure and communicate the objects of OOSE analysis model. From this schema the Use-case-PAC model is defined. The PAC agents will be implemented according to an agent framework defined in [LM 96]. Here, we will describe the OOMGRIN method, using the mentioned results from previous works.

The OOMGRIN Method.

I. *Formulation of the requirements for the application following the OOSE guidelines*. The Requirements model is developed, and concerns the following aspects: the use-case model, where the actors are identified with their functionalities (use-cases) with respect to the system; the description of the system´s interfaces; a description and /or a specification of the use-cases; and finally, an optional object model of the application.

II. *Creation of the Analysis Model, on the basis of the requirements formulated in step I*. This step develops a first logical system structure in which three dimensions are focused: the system behavior, the information contained in the system and the presentation of the system to the exterior world. Three types of objects are identified during this step: *control*, *entity* and *interface objects*. The schema (Figure 2) for organizing these objects developed in Section 4 is used for every "human" actor of the system.

III. *Obtention of the Use-case-PAC model*. The PAC model allows the definition of a system architecture where user-interface aspects are focused. Based on the schema defined in step II, the Use-case-PAC model is developed showing the different levels corresponding to the system´s actors, functionalities requiring user-interface facilities.

IV. *Design of the Use-case-PAC agents using frameworks.*. Two frameworks or semifinished generic architecture are used for developing each interface agent of the Use-case-PAC model. The approach to software design with patterns or templates that can be applied in many situations, captures the experience involved in designing object oriented (OO) software. A framework describing the implementation of a PAC agent has been presented in [LM 96].

V. *Implementation of the classes corresponding to the agents.* Reuse of the available toolkit classes and communication schemas preserving the PAC principles [Los&Al 94a], [Los&Al 94b] are used to implement the graphical (low level) objects.

In what follows we will be discussing in details each one of these steps.

5.1 Formulation of the requirements for the application following the OOSE guidelines.

A requirements model [Jac&Al 92] is created from the requirement specification in which we delimit the system and define all the functionality of the system. Two main elements: *actors* and *use-cases* help in developing this first transformation from the requirement specification. These concepts are simply an aid to defining what exists outside the system (actors) and what should be performed by the system (use cases). The actors represent what interacts with the system and also show everything that needs to exchange information with the system. A use case is a description in which identifies its transactions what are initiated by an actor. After we have defined what is outside our system we can define the functionality inside it. We do this by specifying use cases. A use case is a specific way of using the system by performing some part of the functionality.

The use case model [Jac&Al 92] uses actors and use cases for developing its complete structure. This use case model is part of the requirements model and will show all of the functionality of the system. The use case model will control the formation of all other models (Figure 1) It is developed in cooperation with the domain objet model. It is necesary to describe the interfaces in more detail in order to support the above use case and also the communication between them. Depending on the complexity of the user interface it could be interesting to be careful in describing interface descriptions.

5.2 Creation of the Analysis Model, on the basis of the requirements formulated in step I.

The Analysis Model [Jac&Al 92] aims to structure the system independently of the actual implementation environment. This means that we focus on the logical structure of the system. It is here that we define the stable, robust and maintainable structure that is also extensible. According to the OOSE approach, entity and interface objects should have been identified before the control objects. Control objects are non-persistent, in the sense that they only exist during the use-case execution. Control objects allow also communication among the other objects, connecting different scenarios. A control object can be assigned to each concrete or abstract use-case, where the interface and entity objects express the behaviour that has not been captured in the other objects. When a control object is related to different actors, its behaviour depends on the actor and it should be decomposed into different control objects, one for each actor. This descomposition favors extension and maintenance because modifications are induced by the actors. In general, the kind of funcionalities expressed by a control object are transaction oriented or particular control sequences of one or several use-cases. All the funcionalities specified in the use-case descriptions depending directly from the environment exterior to the system, are expressed by the interface objects. The actors communicate with the system through these objects. The role of an interface object is to translate the actor actions on the system into system events involving the actor and its queries. Interface objects describe bidirectional communications between the system and its users. The functionalities must be distributed into entity, interface and control objects. The schema defined in Figure 2 is used to define and distribute these objects.

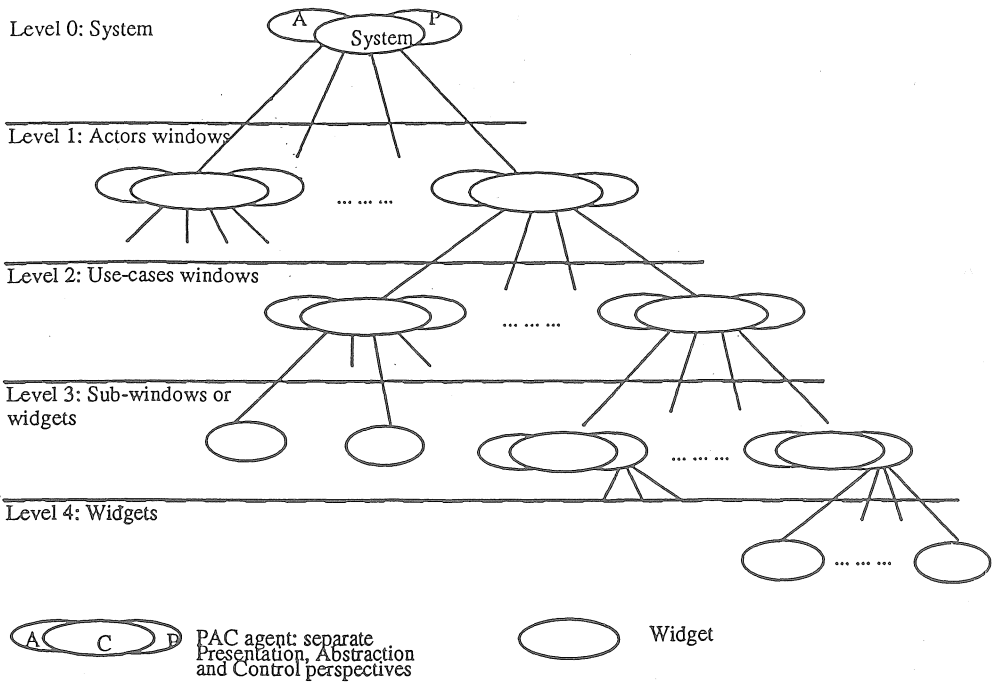
5.3 Obtention of the Use-case-PAC model.

The Use-case-PAC model is obtained through a schema defining the objects in the OOSE analysis model, with respect to a given "human" actor.The PAC model allows the definition of a system architecture where user-interface aspects are focused. In what follows, the PAC model derived from the use-case approach presented in the schema seen in 4, is considered. In the schema, the objects are defined with respect to a given actor of the system. The PAC diagram in Figure 3 shows levels corresponding to the system's actors, having functionalities requiring user-interface facilities.

In the diagram, Levels from 1 to 4 correspond to the different interface objects used for interface design, based on the use-case approach, according to the OOSE method. The schema defined in 4 allows the separation of the application or domain problem component from the interface component, since the only communication is through control objects. At Level 0, the control perspective of the agent called *System*, corresponds to the set of control objects of the OOSE analysis model. The strategy for defining their functionalities is to use balanced control objects. These will communicate with the PAC agents of Level 1 only through their control perspectives. The abstraction perspective of the system agent (*A* in the diagram), corresponds to the problem domain, that is to say to the set of entity objects of the OOSE analysis model. As we have already mentioned, the presentation perspective of this agent (*P* in the diagram) correspond to just an icon to start the application, or a window representing the whole application, from where its functionalities may be accessed.

Notice that in the Use-case-PAC model, the agents are equivalent to the interface objects identified in the OOSE analysis model; this correspondence is possible according to the schema presented in 4. The advantages of the Use-case-PAC model are:

- Structure hierachically the interface objects.
- Decompose each interface object into three perspective (abstraction, presentation, control), with all the advanteges of the PAC agents.
- Provide an object-oriented design for the user-interface component.
- Have a clear separation between the problem domain and the user-interface component in the system architecture.
- Facilitate modifications and the implementation step [Los&Al 94a], [Los&Al 94b].



Level 0: the agent corresponds to the application that is being developed
Level 1: each agent corresponds to a CIO_a , called actor window, from which the actor's use-cases are accessed
Level 2: each agent corresponds to a CIO_{i_a}, 1_i_n, from which the actor can execute functionalities associated to one or more use cases. For each agent (actor window) of Level 1, we have at most n windows.
Level 3: each agent corresponds to non central interface objects, which are windows contained in the actor windows
Level 4: each object corresponds to a widget.

Figure 3. The Use-case-PAC Model

5.4 Design of the Use-case-PAC agents using frameworks..

According to [Pre 95] a framework is defined as a generic semifinished architecture for various application domains. One of the first application of frameworks was in the domain of GUI. Actually, MVC is one of the first known frameworks [KP 88]. A framework is constituted by one or several design patterns [Gam&Al 95], that are used for rapid development of adaptable (reusable and extensible) GUI. A framework represents a generic subsystem that can be instantiated through public abstract classes of the framework (hot spots).

The approach to software design with patterns or templates that can be applied in many different situations, capture the experience involved in designing object-oriented (OO) software. Each design pattern systematically names, explains and evaluates an important and recurring design OO systems. Recurring patterns of classes and communicating objects are found in many object-oriented system. These patterns solve specific design problems and make OO designs more flexible, elegant and reusable. They help designer reuse successful designs by pasing new designs on prior experience. A designer, familiar

with such patterns can apply them inmediatly to design problems without having to rediscover them. Patterns, at a given level of abstraction, are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context. A design pattern names, abstracts and identifies the key aspects of a common design structure that make it useful for creating a reusable OO design.

Considering these ideas, a framework for a PAC agent has been defined in [LM 96]. Three main patterns characterize the PAC model: *Mediator*, *Strategy* and *Factory Method*. The control, which introduces the notion of communication between abstraction and presentation, mediating for the coherence between the two perspectives, is modelled by the *Mediator* pattern. It is important to point out that, since a PAC agent may be decomposed in subagents, due to the PAC agents hierarchy, each subagent has its own control, and the nested views are treated as subagents. The *Strategy* pattern attaches a view to a controller allowing to change the way a view responds to user inputs and finally the *Factory Method* relates a view with the different incapsulated treatements of the user inputs, also like MVC.

5.5 Implementation of the classes corresponding to the agents.

Depending on the multiagent model, we could select an appropiate toolkit available in the market. When the toolkit has tools at a high abstraction level, the tools have more control on the agent than the user or implementor; this mean that the presentation and the control are kept together and the two perspectives cannot be separated as for PAC agents [LG 95]. This is the case of MVC like agents, where the controller and the view are together in the presentation perspective, and the abstraction is separated from the rest of the agent and the case of agents with non separated perspectives. This combination of models seems well adapted to toolkits with tool at lower abstraction level, like X,Xt[3]/Motif[4]. The implementor has more control on the agents and three separate perspectives can be always considered. An important issue is that at application level, an abstraction perspective is maintained in both models, and at interface level, the lower level the specific application component objects have also a separate abstraction. This fact guarantees that the PAC principles are maintained in the sense that this architecture favors the distribution of semantical and syntactical actions, enhancing the system flexibility to changes. The communications among the PAC agents follows a schema defined in [Los&Al 94a], [Los&Al 94b] which preserves the PAC principles, to implement directly the communications in C++ [Str 93]. The three PAC perspectives are builted as separate classes.

6. CONCLUSIONS.

This work describes an OO method for developing interactive systems. Our method enhances the early separation of GUI aspects from the semantic application features, favoring also a scenarios-oriented design. Notice that steps I, II, III correspond to the analysis phase, that it is performed using the Use-case-PAC model. Steps IV and V correspond to the detailed design phase. Particular care has to be taken for the development of the abstraction corresponding to the application level agent, which represents the

[3] X-Window is registered trademark of the Massachusetts Institute of Technology.
[4] Motif is a registered trademark of Open Software Foundation.

problem domain component and can be modelled accordingly, using any of the existing OO methods, or a combination of them, within the spirit of recent trends in software development methodology. We claim [LG 95] that the structure of the PAC agents may be greatly influenced, according to the choices of the plataform and the graphical toolkits. In the sense of that, according to the toolkit tool, the control perspective of an agent cannot be always separated from the presentation and abstraction perspective. Nevertheless, the PAC communication principles can still be preserved. The guidelines for interactive software development presented in this paper have been used successfully in several software projects at the ISYS[5] Research Center. The benefits from this experience with respect to groupware are multiple, in the sense that the work can be easily distributed among the programmers' teams, assigning to different teams the separate development of the presentation and the abstraction perspectives of the agents constiting the GUI of the application. Finally, another different team can be in charge to implement the predefined communication schema [Los&Al 94a], [Los&Al 94b] between agents and their perspectives. Moreover, the problem domain component can be developed in parallel with the GUI component, favoring an early prototyping of the application. The formalization of the concepts underlying the OOMGRIN method and the experimentation with the system's adaptability to changes in the GUI, as for example the incorporation of different interaction media such as voice and videos are undergoing research topics.

7. REFERENCES.

[Car 95] CARROL J., M., "Scenario-Based Design", Jhon Wiley &Sons Inc., 1995.

[Cic 84] CICCARELLI E. C., "Presentation Based User-Interfaces", Technical Report 794, Artificial Intelligence Laboratory, Massachussets Intelligence Laboratory, August 1984.

[Col 95] COLLINS D., "Designing Object-Oriented User Interfaces", Benjamin/Cummings Publishing Company, Inc., 1995.

[Cou 90] COUTAZ J., "Interfaces homme-ordinateur", Dunod 1990.

[Del 92] DELMAS S. "Tcl/TK User's Manual", University of California, Berkeley, 1992.

[Jac&Al 92] JACOBSON I., CHRISTERSON M, JONSSON P., ÖVERGAARD G., "Object-Oriented Software Engineering, a Use Case Driven Approach", Addison Wesley, 1992.

[F&D 84] FOLEY D.J., VAN DAM, "Fundamentals of Interactive Computer Graphics", Addison Wesley, 1984.

[Gar 93] GARFINKEL S.L., MICHEL K.M. "The NeXSTEP Programming: Step One, Object-Oriented Applications" A Tutorial for developers using the Objective-C Language and the NeXT Interface Builder. Santa Clara, CA:Telos/Springer-Verlag, 1993.

[Gol 84] GOLDBERG A., "Smalltalk-80 - The Interactive Programming Environment", Addison-Wesley 1984.

[HH 89] HARTSON R., HIX, D., "Human-Computer Interface Development Concepts and Systems for its Management", ACM Computing Surveys, Vol.21, No.1, March 1989.

[Lan 86] LANTZ K. A., "On User-Interface Reference Models", ACM SIGCHI, Bulletin, vol. 18, 2 (1986), 36-44.

[LG 95] LOSAVIO F., GALVEZ C.,"The platform influence on object-oriented development based on the multiagent model", Proceedings of the XXI Conferencia Latinoamericana de Informática, Panel'95, Canela RS, Brasil, August 1995.

[5] Centro de Ingeniería de Software y Sistemas.

[Los 95] LOSAVIO F. "An Object-Oriented Methodology for User-Interface Design Based on the Multiagent Model", Proceedings of the Information Systems Analysis Synthesis, ISAS'95, 89-92, Baden-Baden, Germany, August 1995.

[Los&Al 94a] LOSAVIO F., MATTEO A., ORDAZ O., MEZA O., GONTIER W. "An implementation of the PAC architecture using object-oriented techniques" Proceedings IFIP'94, 13th World Computer Congress 94, Volume 2, 149-155. Hamburgo-Alemania, Agosto 1994, K. Brunnstein and E. Raubold (Editors) Elsevier Science B.V. (North-Holland).

[Los&Al 94b] LOSAVIO F., MATTEO A., ORDAZ O., MEZA O., GONTIER W. "Object-oriented approach and PAC model: design of the GReAt (Graph Researcher Assistant) environment" Proceedings XX Conferencia Latinoamericana de Informática, Panel'94, 433-443, Monterrey, Estado de México, Septiembre 1994.

[LM 95] LOSAVIO F, MATTEO A., " Use-case and Multiagent Models for Object-Oriented Design of User-Interfaces", L.R.I., Orsay, France, Rapport de Recherche No. 992, Sept. 95. To appear in Journal of Object- Oriented Programming.

[LM 96] LOSAVIO F, MATTEO A., "Object-Oriented User-Interface Design Based On Agents Frameworks", Centro ISYS, Reporte de Investigación, ISYS No. 1, RI/01/96, Enero 1996.

[LVC 89] LINTON M.A., VLISSIDES J.M., CLADER V., "Composing User-interfaces with Interviews", IEEE, Computer, 8-22, Feb. 1989.

[Mey 88] MEYER B., "Object-Oriented Software Construction", Prentice Hall 1988.

[Ste 87] STERN H. L. , "Comparison of Window Systems" BYTE 12, 11, Nov. 1987.

[Str 93] STROUSTRUP B., "The C++ Programming Language", Second Edition, Addison-Wesley Publishing Company, 1993.

[KP 88] KRASNER G. E., POPE S.T. " A Cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80" Journal of Object-Oriented Programming 1(3), 1988.

[Pre 95] PREE W. "Design Patterns for Object-Oriented Software Development", Addison Wesley, 1994.

[Gam &Al 95] GAMMA E., HELM R., JOHNSON R.,VLISSIDES J. "Design Patterns. Elements of Reusable Object-Oriented Software" Addison Wesley Publishing Co., 1994.

Un lenguaje de programación orientado a objetos activos

Claudia Pons Gabriel Baum

LIFIA, Universidad Nacional de La Plata, Argentina.
cpons@ada.info.unlp.edu.ar

Keywords: programación orientada a objetos, sistemas reactivos, lenguajes de programación, semántica formal.

Resumen

En este artículo definimos las bases para un lenguaje de programación orientado a objetos que permite expresar comportamiento activo en forma unificada. El comportamiento activo queda embebido dentro del comportamiento general de los objetos del sistema. Se respeta el encapsulamiento de los objetos y se aprovecha la jerarquía de clases para determinar el alcance de las reglas.

El comportamiento del sistema se va definiendo a lo largo de la ejecución, ya que cada objeto al desplegar su comportamiento (ejecutar un método) puede agregar y/o borrar mensajes activos del sistema. Es decir, cada objeto tiene la posibilidad de "legislar" (dictando y/o anulando reglas.) sobre el sistema. El lenguaje permite expresar distintas formas de comportamiento activo: general, particular, persistente y efímero.

1. Introducción

La crisis del software que se desencadenó en las últimas décadas obligó a los investigadores a buscar nuevas metodologías de trabajo para aumentar la productividad de los desarrolladores de software.

Se trabajó sobre el principio de que una buena metodologías de desarrollo de soft debe proveer como mínimo: • *modularidad*: dominio de la complejidad de los grandes sistemas mediante su adecuado particionamiento. • *reuso*: una pieza de soft definida para un contexto puede ser reutilizada en otro contexto diferente. • *mantenibilidad*: cambios en el sistema pueden ser localizados e implementados en forma fácil y económica.

En este contexto surgió el paradigma de orientación a objetos que permite modelar sistemas en términos de objetos, lo cual favorece la producción de programas mas legibles, reusables y mantenibles. El nuevo paradigma logró una rápida aplicación comercial mediante lenguajes como Smalltalk, Eiffel y C++.

Paralelamente a los sistemas orientados a objetos también cobraron importancia los sistemas basados en conocimiento y sistemas expertos. Emergió un nuevo estilo de programación - basada en reglas- que encuentra su expresión en lenguajes tales como Prolog y OPS5.

Hoy en día, tanto programación orientada a objetos como programación basada en reglas son paradigmas populares e igualmente aceptados. No es posible determinar cual de los dos paradigmas es mas adecuado, ya que esto depende de la tarea que se intente codificar. Sin embargo existen casos donde sería deseable poder usar ambos paradigmas en forma combinada, ya que ninguno de ellos aisladamente resulta de óptima aplicabilidad.

Básicamente existen dos formas de integrar sistemas orientados a objetos con sistemas basados en reglas:

- una posibilidad es permitir al programador incluir llamadas a un programa orientado a objetos dentro de un sistema basado en reglas. Sin embargo, con esta solución se pierden las principales ventajas de la programación orientada a objetos, ya que los lenguajes basados en reglas existentes en general no proveen encapsulamiento y modularidad.
- la otra alternativa es extender un lenguaje orientado a objetos para soportar reglas. Esto permite conservar todas las ventajas de la programación orientada a objetos y además ganar mecanismos declarativos para especificar restricciones y comportamiento guiado por eventos (es decir, comportamiento activo), tanto locales a un objeto como globales a un grupo de objetos.

Existen varias propuestas para integrar conceptos activos con conceptos orientados a objetos, por ejemplo: el sistema de reglas CERS-rules system [Miranker93] que soporta reglas de producción OPS5 dentro de un ambiente C++; el sistema TANGUY [Eich93] propone una extensión del lenguaje C++ para soportar reglas de producción reactivas (que reaccionan a eventos que ocurren durante la ejecución de un sistema, tales como envío de mensajes o cambios en los datos de objetos particulares). El sistema Opus [Atkinson87] y su extensión el sistema NéOpus [Pachet95] permiten embeber reglas de producción dentro del lenguaje Smalltalk-80 La integración de los dos paradigmas también ha recibido gran atención en el área de las bases de datos [Gatziu91, Gehani91, Medeiros91, Kappel94, Bertino94]

2. Planteo del Problema

Las soluciones existentes permiten integrar los dos paradigmas. Los resultados son sistemas híbridos consistentes de dos lenguajes diferentes, uno para los objetos y otro para las reglas. En estos sistemas los paradigmas conviven pero no se unifican. Por ejemplo, en Tanguy las clases se definen en C++, mientras que el comportamiento activo se define a través de reglas para las cuales se provee una sintaxis especial. Los diseñadores deben definir el comportamiento activo separadamente y no en forma integrada con las clases del sistema. En TriGS [Kappel94] las reglas activas y los eventos que hacen que esas reglas se disparen son parte de la jerarquía de clases de la aplicación. Esto conduce a una modelización donde un evento es un objeto de

primera clase. Sin embargo, conceptualmente un evento es un cambio de estado de un objeto y no un objeto en sí mismo. NéOpus elimina el "impedance mismatch" entre los lenguajes permitiendo expresar las reglas en términos de Smalltalk. Sin embargo, las reglas no están embebidas dentro del comportamiento de los objetos, sino que ellas constituyen en sí mismas un objeto distinguido dentro del sistema que gobiernan.

Nuestra propuesta tiene por objetivo unificar los paradigmas de manera que el comportamiento activo quede embebido dentro del comportamiento general de los objetos del sistema. Deseamos que los desarrolladores del sistema tengan un marco de trabajo integral, donde puedan expresar todo el comportamiento de los objetos (ya sea activo o pasivo) dentro del mismo lenguaje y al mismo nivel. Para esto definimos un prototipo de lenguaje unificado y damos su semántica formal. En nuestro lenguaje el comportamiento activo se modeliza a través de diferentes tipos de reglas (o mensajes activos) escritas en el mismo lenguaje. Estas reglas tienen la misma jerarquía que los mensajes comunes del sistema y forman parte del comportamiento de los objetos.

3. Taxonomía de Mensajes en los Sistemas Orientados a Objetos

En los sistemas orientados a objetos un grupo de objetos interactúa para resolver un problema. La única forma de interacción es el envío de mensajes. Cada clase de objeto se caracteriza por el conjunto de mensajes que sus objetos pueden aceptar. Dado que las clases son el único ámbito para la definición de mensajes, resulta claro que todo mensaje que circula por el sistema tiene un objeto receptor. Sin embargo no todo mensaje que circula por el sistema tiene un objeto emisor (existen mensajes cuyo emisor no es un objeto). Basándonos en su forma de emisión clasificaremos a los mensajes que circulan por el sistema en distintas categorías

3.1. Los mensajes en un Sistema Orientado a Objetos Clásico

En un sistema orientado a objetos clásico (como C++ o Smalltalk) los objetos sólo reaccionan al recibir un mensaje, por lo tanto un objeto sólo puede enviar un mensaje como respuesta (reacción) a otro mensaje. Existen además mensajes contenidos en el programa principal (Main). El emisor de estos mensajes no es ningún objeto del sistema. Por lo tanto, en estos sistemas distinguiremos dos categorías de mensajes (fig.3.a):

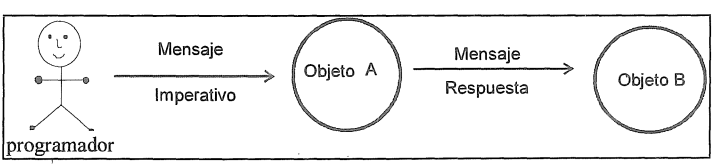


Fig.3.a Mensajes en un Sistema Orientado a Objetos.

Mensajes Imperativos: son los mensajes del programa principal, su emisor no es ningún objeto del sistema.

Mensajes Respuesta: su emisor es algún objeto del sistema, es decir son los mensajes que algún objeto envía como respuesta a otro mensaje.

3.2. Los mensajes en un Sistema Orientado a Objetos Activos

Los sistemas orientados a objetos tradicionales no son adecuados para representar comportamiento guiado por eventos; esto se debe a que los objetos tradicionales sólo reaccionan al recibir un mensaje y no pueden reaccionar automáticamente ante determinados cambios de estado del sistema. Es decir, los objetos tradicionales sólo responden a ordenes, no poseen capacidad para iniciar acciones por decisión propia. En consecuencia, para modelar comportamiento activo es necesario agregar una nueva categoría de mensajes:

Mensajes Activos: los mensajes activos no tienen un objeto emisor, se disparan automáticamente en respuesta a algún evento, es decir, cuando el sistema alcanza algún estado particular. La figura 3.b muestra la evolución de un evento dentro de un sistema activo.

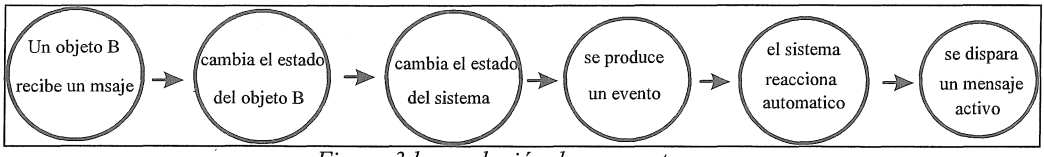


Figura 3.b: evolución de un evento

Utilizaremos la palabra **regla** para referirnos a estos mensajes activos. Una regla tiene la forma < **conditions** Exp **actions** Com>, siendo Exp una expresión que representa las condiciones que deben cumplirse para que ocurra el evento. Com es un comando que representa las acciones que se desencadenan en respuesta al evento.

Clasificación de los Mensajes Activos

Distinguimos diferentes categorías de reglas teniendo en cuenta su alcance con respecto a los objetos que involucra y con respecto a su tiempo de vida:

1 - <u>De acuerdo a los objetos que involucra</u>:

una regla puede referirse a un grupo particular de objetos o puede referirse a objetos genéricos

<u>Categoría 1.1</u>: ***Reglas Generales***

involucran un conjunto genérico de individuos. Por ejemplo, en un cierto Banco la política financiera es clausurar las cuentas cuando su saldo es inferior a un monto mínimo establecido. Esto puede representarse con la siguiente regla, válida para toda instancia c de la clase CuentaBancaria

Rule "Cuentas en Rojo":
conditions: la cuenta c está activa y el saldo de c es inferior al mínimo
actions: clausurar la cuenta c.

Categoría 1.2: ***Reglas Particulares***

involucran un conjunto constante de individuos. Por ejemplo: Mr.Perez es un cliente especial del Banco al cual se le efectúan préstamos automáticos cuando desciende el saldo de su cuenta, entonces para él definimos la siguiente regla:

Rule "Rojos de Mr.Perez":
conditions la cuenta de Mr.Perez está activa y su saldo es inferior al mínimo
actions depositar $10000 en la cuenta de Mr.Perez .

2 - De acuerdo a su tiempo de vida:

Categoría 2.1: ***Reglas Persistentes***

pueden ejecutarse múltiples veces (modelan eventos que se repiten). Por ejemplo las reglas "Cuentas en Rojo" y "Rojos de Mr.Perez".

Categoría 2.2: ***Reglas Transitorias***

se ejecutan sólo una vez y luego desaparecen (modelan un evento que ocurre sólo una vez). En la sección 6 describimos un ejemplo que incluye este tipo de reglas.

4. Dominios Semánticos para Objetos

En esta sección definimos dominios para interpretar objetos activos. Estos dominios brindarán la base para expresar la semántica de los programas que manipulan objetos.

- **Object = (Bvalue x Behavior x List_Rule)** , dominio para objetos.
- Bvalue, dominio primitivo para valores básicos (ej: bool, nat).
- Behavior=[Message → Method] dominio para comportamiento de objetos. Un comportamiento es una tabla que asocia cada mensaje con el método que el objeto computará para responder a ese mensaje.
- Message = String, dominio primitivo para nombres de mensajes. Este dominio está particionado en Observer_Message y Mutator_Message..
- Method = Exp + Com, dominio de métodos. Un método puede ser una expresión (método observador) o un comando (método mutador).
- List_Rule, dominio para listas de reglas.

Fig. 4.a -Dominio semántico para Objetos

El dominio donde los objetos toman significado es el conjunto de **registros** en forma similar a lo propuesto por Cardelli en [Cardelli85]. Más precisamente, modelamos un objeto activo como una terna, donde la primera componente es el estado interno, la segunda es el comportamiento

pasivo (el conjunto de todos los métodos que el objeto puede ejecutar (propios y heredados)) y la tercera son las reglas particulares que determinan su comportamiento activo individual. Consideramos que el conjunto de métodos de un objeto está particionado en dos subgrupos: *observadores* y *mutadores*. Un observador retorna un resultado sin alterar el estado interno de ningún objeto. Un mutador modifica uno o más estados de objetos. La Figura 4.*a* muestra la definición del dominio semántico Object. En un sistema orientado a objetos cada objeto posee un identificador único (generalmente denominado oid). Sin embargo, estos identificadores no son visibles y los programas manipulan los objetos mediante nombres simbólicos (o variables). En la figura 4.b definimos los dominios semánticos Env y Store. Un ambiente (environment) es una función que asocia nombres simbólicos con valores expresables. Un valor expresable puede ser un valor básico o un identificador de objeto. Estas funciones no son necesariamente inyectivas, lo cual permite que en un programa dos o mas nombres referencien al mismo objeto. Un almacenamiento(store) es una función que asocia oids con objetos. Estas funciones son inyectivas garantizando que cada objeto posee un único oid.

- Oids, dominio primitivo para identificadores de objetos.
- Vars, dominio primitivo para nombres de variables.
- Evalue = Bvalue + Oids, dominio para valores expresables.
- **Env = [Vars → Evalue]** dominio para ambientes de variables. Un ambiente es una función que liga nombres de variables con valores expresables.
- **Store = [Oids → Object]** dominio para almacenamientos de objetos. Un store es una función inyectiva que liga identificadores de objetos con objetos.

Fig. 4.b -Dominios semánticos para Sistemas de Objetos.

5. Un lenguaje de programación orientado a objetos activos

En esta sección consideramos un simple lenguaje de programación que provee las construcciones mínimas para soportar programación orientada a objetos (tal como definición de clases y envío de mensajes en paralelo) y además permite expresar *mensajes activos* (reglas) dentro de un marco uniforme con los restantes mensajes del sistema.

Sintaxis:

1- Categorías Sintácticas:

D in Declarations
meth in Meth
o in Observer_Message
m in Mutator_Message
bv in Bvalue
s in String
ch in Channel
e in Exp
c in Com p in Prog
list in List f in Fun
r in Rule x in Vars

2. Definiciones:

D ::= **Class** s **Methods** list_meth **Rules** list_r

meth ::= **obs_method** o(x) e | **mut_meth** m(x) c

e::= bv | f(e_1,..., e_k) | x| *state*(x) | e_1.o(e_2)

c ::= **skip** | *update*(x,e) | ch?x | ch!e | x:= e | x:=**new**(s)
e_1.**m**(e_2) | c_1;c_2 | c_1||c_2 | **insRule**(r) | **delRule**(r)

r::= (**conditions** e **actions** c)

p ::= **Program** list_D c

Utilizamos negrita para indicar los símbolos terminales del lenguaje (por ejemplo **Class**, **Methods, skip**). List_x es una abreviatura genérica que representa listas finitas de elementos de tipo x, es decir, list_x:: = x | x:list_x
Un programa en este lenguaje incluye declaraciones de clases activas y comandos. La declaración de una clase incluye un nombre para la clase y una secuencia de métodos (que representan el comportamiento pasivo tradicional de los objetos). Cada método puede ser un observador o un mutador. El cuerpo de un método observador es una expresión, mientras que el cuerpo de un método mutador es un comando. La declaración de una clase también incluye una lista de **reglas generales**, aplicables a todas las instancias de la clase. Las **reglas particulares** no se declaran sino que se crean dinámicamente en tiempo de ejecución.
Las expresiones comprenden: - valores básicos,- términos de función, - las variables (incluyendo a la variable especial *self* que representa al objeto en sí mismo), - el término *state*(x) que representa el estado interno oculto del objeto ligado a la variable x, - el término $e_1.o(e_2)$ que representa el envío de un mensaje observador con parámetro e_2 al objeto resultante de evaluar la expresión e_1.
Los comandos incluyen: - skip, el comando que no realiza ninguna acción, - el comando *update*(x,e) que representa la modificación del estado interno del objeto ligado a la variable x, asignándole la expresión e (en programas sintácticamente correctos la única variable que puede ocurrir como parámetro de *state* y *update* es la variable especial *self*) , - entrada y salida de datos, - asignación, - *new*(s) que simboliza la creación de un nuevo objeto de clase s, - envío de un mensaje mutador, - el operador ; que indica composición secuencial de comandos, - el operador || que indica composición paralela, - **insRule**(r) para agregar una nueva regla particular en el sistema y **delRule**(r) para eliminar una regla particular.

Reglas Particulares y Reglas Generales

El lenguaje permite estructurar las reglas de la siguiente forma: las reglas generales están asociadas con las declaraciones de clases y las reglas particulares quedan encapsuladas dentro de los objetos, como se muestra en la figura 5.a.. El alcance de las reglas (individuos sobre los cuales cada regla es potencialmente aplicable) está determinado por la jerarquía de clases del sistema. Una regla general es aplicable sobre todas las instancias de la clase donde está definida y también sobre todas las instancias de sus subclases. Evaluar la condición de una regla general tiene complejidad de O(n), siendo n la cantidad de objetos en la clase (incluyendo subclases). Los reglas particulares se refieren a objetos individuales. La evaluación de la condición de una regla particular tiene complejidad de O(1).

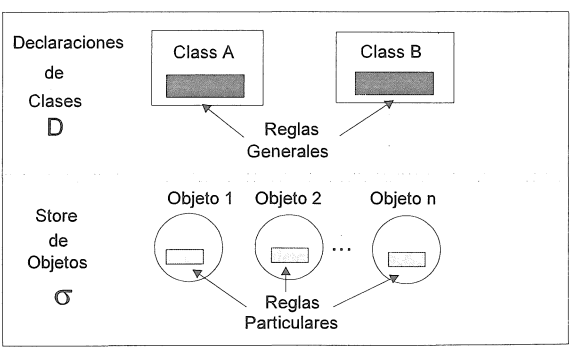


Fig 5.a: Distribución de las reglas dentro del sistema.

El comportamiento del sistema se va definiendo a lo largo de la ejecución, ya que cada objeto al desplegar su comportamiento (ejecutar un método) puede agregar y/o borrar mensajes activos del

sistema. Es decir, cada objeto tiene la posibilidad de "legislar" sobre el sistema (dictando y/o anulando reglas).

Semántica:

Utilizaremos conceptos del formalismo de semántica operacional estructural [Hennessy90].

Las variables δ, σ, bv, ev, obj y oid denotan elementos de los conjuntos Env, Store, Bvalue, Evalue, Object y Oids respectivamente.

Semántica de las expresiones :

Las expresiones denotan valores básicos o identificadores de objetos (llamamos valores expresables, Evalue, al dominio semántico de las expresiones).

Asumimos que no es de interés observar el proceso de computación de expresiones; por lo tanto definimos una función que asocia directamente cada expresión con el valor final que esta denota.

Definimos la función $\Rightarrow_e$ mediante reglas, de la siguiente forma:

$$\Rightarrow_e :: < \text{Exp} , \text{Env} , \text{Store} > \rightarrow <\text{Evalue} >$$

Regla BR

$$(bv,\delta,\sigma) \Rightarrow_e bv$$

Regla VarR

$$(x,\delta,\sigma) \Rightarrow_e \delta(x)$$

Regla StateR

$$(state(x),\delta,\sigma) \Rightarrow_e \Pi_1(\sigma\ (\delta(x)))$$

La regla BR indica el significado de un valor básico. La regla VarR se refiere a la evaluación de variables. De acuerdo con la regla StateR, state(x) denota el estado interno del objeto que en el momento de la evaluación se halla ligado a la variable x. En nuestro dominio los objetos son ternas. Π_1 representa la primera proyección de la terna, es decir el estado interno del objeto.

Regla FunR

$$\frac{(e_1,\delta,\sigma) \Rightarrow_e bv_1 \quad \ldots\ldots\ldots \quad (e_k,\delta,\sigma) \Rightarrow_e bv_k}{(f(_k,\ldots, e_k),\delta,\sigma) \Rightarrow_e Apply(f, bv_1, \ldots, bv_k)}$$

Regla ObsR

$$\frac{(e_1,\delta,\sigma) \Rightarrow_e oid \quad (e_2,\delta,\sigma) \Rightarrow_e ev \quad (B(o), \delta[oid/self]\ [ev/x],\sigma) \Rightarrow_e ev'}{(e_1.o(e_2),\delta,\sigma) \Rightarrow_e ev'}$$

donde, $B = \Pi_2(\sigma(oid))$.

FunR describe la evaluación de funciones primitivas (tal como suma de enteros).

ObsR describe la evaluación de un mensaje observador. La expresión e_1 denota al identificador del objeto receptor del mensaje o. Llamamos B al comportamiento de este objeto. Entonces B(o) es el método que el objeto computa como respuesta al mensaje o. Para la evaluación del método (que es una expresión) la variable especial self se instancia con el objeto receptor y el parámetro formal x se instancia con el parámetro real ev.

Semántica de los comandos :

Llamamos **Act** al conjunto de acciones primitivas,
Act= { c?bv / c∈Chan , bv∈Bvalue } ∪ { c!bv / c∈Chan , bv∈Bvalue } ∪ { ε }
Este conjunto está integrado por las acciones de entrada y salida de datos y por la acción interna ε. Una acción interna es una acción no observable desde el exterior, por ejemplo las modificación del store o la comunicación interna de dos comandos corriendo en paralelo.

La semántica de computación de nuestro lenguaje está dada por las relaciones $\stackrel{a}{\Rightarrow}c$, una para cada acción ***a***, de tipo $\stackrel{a}{\Rightarrow}c$ ::: <Com,Env, Store> → <Com,Env, Store>,

donde la sentencia (c,δ, σ) $\stackrel{a}{\Rightarrow}c$ (c',δ', σ') significa que el comando c puede ejecutar una acción primitiva ***a*** sobre el ambiente δ y el store σ transformándolos en δ' y en σ' respectivamente, quedando en c' lo que aún resta por ejecutar del comando c. Además, para dar el significado del lenguaje necesitamos tener en cuenta las declaraciones de clases definidas por el usuario; concretamente vamos a definir relaciones parametrizadas sobre declaraciones de clases.

Las reglas para comandos usan un predicado de terminación ✓, cuya definición es la clásica [Hennessy90]. Por limitaciones de espacio sólo presentamos algunas de las reglas que definen la semántica de los comandos.

Regla InR

$$D \vdash (ch?x,\delta,\sigma) \stackrel{ch?v}{\Rightarrow} c(skip, \delta[bv/x] , \sigma)$$

La regla InR describe el siginificado del comando ch?x que permite leer del canal ch y colocar el valor leído dentro de la variable x. El store de objetos no se modifica.

Regla NewR

$$D \vdash (x:= new(s),\delta,\sigma) \stackrel{\varepsilon}{\Rightarrow} c(skip,\ \delta[oid/x] , \sigma[obj/oid])$$

donde, existe en D una declaración de clase de la forma **Class** s **Methods** m **Rules** r

La regla NewR denota la creación de un nuevo objeto. La función next_oid retorna un identificador no usado dentro del store, sea oid= next_oid(σ). Este identificador se liga a la variable x dentro del ambiente. La función Sem construye una nueva instancia de la clase s a partir de su declaración. Su tipo es Sem:: Declarations → Object. Para una descripción detallada de la función (incluyendo el tratamiento de definiciones de clases utilizando herencia) puede consultarse [Pons95]. Sea obj = Sem(Class s Methods m Rules r) esa nueva instancia, obj se liga al nuevo identificador dentro del store.

Regla Par2aR

$$\frac{D \vdash (c_1,\delta,\sigma) \stackrel{ch?v}{\Rightarrow} c\ (c'_1,\delta',\sigma') \qquad D \vdash (c_2,\delta,\sigma) \stackrel{ch!v}{\Rightarrow} c\ (c'_2,\delta,\sigma)}{D \vdash (c_1 \| c_2,\delta,\sigma) \stackrel{\varepsilon}{\Rightarrow} c\ (c'_1 \| c'_2,\ \delta',\sigma')}$$

Regla Par1aR

$$\frac{D \vdash (c_1,\delta,\sigma) \stackrel{a}{\Rightarrow} c(c',\delta',\sigma')}{D \vdash (c_1 \| c_2,\delta,\sigma) \stackrel{a}{\Rightarrow} c(c' \| c_2,\delta',\sigma')}$$

Las Reglas **Par1aR, Par2aR** (y sus dos reglas simétricas Par1bR y Par2bR) definen la semántica de la composición paralela. Los comandos c y c' actúan en forma independiente intercalando sus acciones primitivas. La regla Par2R permite comunicación entre comandos. Si c_1 puede recibir un valor del canal ch y c_2 puede enviar el mismo valor al mismo canal, entonces c_1 y c_2 pueden comunicarse (comunicación handshake). Esta comunicación es una acción interna.

Regla InsR

$D \vdash (insRule(r),\delta,\sigma) \overset{\varepsilon}{\Rightarrow} c(skip, \delta', \sigma')$
donde $< c, \delta' > = const(r, \delta)$; $\sigma' = insert(\sigma, \delta', c)$

La regla InsR define la creación de una nueva regla particular. La regla r, por ser particular, involucra a objetos individuales. Las variables contenidas en r en el momento de la creación de la regla denotan a estos individuos. Dado que las variables pueden ser modificadas (por asignaciones posteriores a la creación de la regla y anteriores a su ejecución), resulta necesario reemplazarlas por constantes. Esto es realizado por la función const. La función insert agrega la regla constante en cada uno de los objetos involucrados en ésta.

Regla delR

$D \vdash (delRule(r),\delta,\sigma) \overset{\varepsilon}{\Rightarrow} c(skip, \delta, \sigma')$
donde $\sigma' = delete(\sigma, r)$

La regla delR indica el significado del comando delRule que elimina una regla particular del sistema. La función delete elimina la regla r de los objetos correspondientes dentro del almacenamiento σ .

Semántica de los programas :

La semántica de computación de los programas está dada por las relaciones $\overset{\lambda}{\Rightarrow} P$, una para cada secuencia de acciones primitivas λ, con λ en Act*. Estas relaciones están definidas en la figura 5.b mediante la regla ProgramR.

Notación : $P \overset{\lambda^*}{\Rightarrow} Q$ si $\lambda = a_1 a_2 ... a_n$ y $P \overset{a_1}{\Rightarrow} p_1 \overset{a_2}{\Rightarrow} p_2 \overset{a_3}{\Rightarrow} ... p_{n-1}. \overset{a_n}{\Rightarrow} Q$

$\overset{\lambda}{\Rightarrow} P :: < Prog > \rightarrow < Env , Store >$

Regla ProgramR

$D \vdash (c , \delta_0 , \sigma_0) \overset{\lambda_1^*}{\Rightarrow} c(c', \delta, \sigma)$
(c', δ, σ) ✓
$D \vdash (\delta, \sigma) \overset{\lambda_2^*}{\Rightarrow} R(\delta', \sigma')$
$D \vdash (\delta', \sigma')$ end

$< Active_Program\ AD\ c > \overset{\lambda}{\Rightarrow} P(\delta', \sigma')$

donde λ es la concatenación de λ_1 y λ_2

Figura 5.b : Semántica de los programas

$\overset{\lambda}{\Rightarrow} R :: < Env , Store > \rightarrow < Env , Store >$

Regla ForwardR

$D \vdash (\delta, \sigma)$ choose (r, δ', σ')
$D \vdash (c, \delta', \sigma') \overset{\lambda_1^*}{\Rightarrow} c (c' , \delta'', \sigma'')$
(c', δ'', σ'') ✓

$D \vdash (\delta, \sigma) \overset{\lambda}{\Rightarrow} R(\delta'', \sigma'')$

La regla seleccionada r es de la forma (**conditions** e **actions** c)

Figura 5.c.: La relación $\overset{\lambda}{\Rightarrow} R$,

La Regla ProgramR indica que el programa comienza ejecutando los comandos contenidos en el main a partir de un almacenamiento inicial σ_0 y un ambiente inicial δ_0. Generalmente estos comandos crearán los objetos del sistema, dictarán algunas reglas particulares y enviarán mensajes imperativos a los objetos para que comiencen a desplegar sus actividades.

Luego de ejecutado el main, el sistema evoluciona al estilo de un sistema de producción. Cada paso de computación involucra la selección de una regla (general o particular) y su ejecución. El

programa termina cuando ninguna regla posee una condición que evalúa a true. Esto es indicado por el predicado de terminación "end" (que por limitaciones de espacio no describiremos aqui).

La relación $\overset{\lambda}{\Rightarrow}R$ representa un paso de computación. Cada paso de computación consiste en la selección de una regla y su ejecución completa (hasta llegar a un estado donde el predicado de terminación ✓ es verdadero).

La relación $\overset{\lambda}{\Rightarrow}R$, parametrizada por las declaraciones, está definida en la figura 5.c.

La relación choose retorna una regla cuya condición evalúa a true. Presenta un comportamiento no determinístico ya que más de una regla puede evaluar a true. Choose retorna además el ambiente conteniendo una posible instanciación de las variables que ocurren en la regla y retorna el store modificado cuando la regla es transitoria. Cuando la regla seleccionada es particular el ambiente no se modifica ya que la regla no contiene variables y cuando la regla seleccionada es persistente el store no se modifica ya que la regla permanece almacenada.

6. Un Ejemplo

Cuando un cliente de un Banco se coloca en la cola de espera de un cajero, decide (en función de su paciencia y tiempo disponible) cuanto tiempo permanecerá en la cola. Si luego de transcurrido ese tiempo el cliente aún no ha sido atendido, renunciará a la espera abandonando la cola.
Este ejemplo describe un evento, que llamaremos ***TopeDeEspera***. Este evento desencadena un mensaje activo que llamaremos ***Impacientarse***. El evento ***TopeDeEspera*** se produce cuando la siguiente condición se vuelve verdadera: " Ha transcurrido al tiempo límite de permanencia del cliente en la cola de espera". El mensaje activo ***Impacientarse***, cuyo objeto receptor es una instancia de la clase Cliente, provoca que el objeto abandone la Cola de Espera.

Implementación en NéOpus

El comportamiento reactivo de nuestro ejemplo se puede modelar con la regla TopeDeEspera de la fig 6.a. NéOpus evalúa esta regla para todas las instancias de la clase cliente, estén o no colocados en la cola de espera; por ello resulta necesario colocar en la regla la condición de que el cliente C pertenezca a la cola Q.

```
Regla TopeDeEspera
| Cliente C. ColaDeEspera Q |
conditions
Q Pertenece (C)
C HoraEncolado + C TiempoLimite > = Reloj
HoraActual
actions
C Impacientarse(Q)
```

Figura 6.a: Regla en NéOpus

Implementación en nuestro lenguaje unificado

Al ingresar al banco el cliente recibe el mensaje Encólese. En respuesta a este mensaje realiza las siguientes acciones: a) se coloca en la cola de espera, b) genera una regla transitoria para modelar el evento potencial de que decida abandonar la cola antes de ser atendido. El mensaje activo ***Impacientarse*** queda

```
mut_meth Encólese ( Q : ColaDeEspera)
   Q.Encolar (self);
   self.HoraEncolado := Reloj.HoraActual ;
   insRule ( conditions self.HoraEncolado +
             self. TiempoLimite = Reloj.HoraActual
             actions self.Impacientarse(Q ) );

mut_meth Impacientarse(Q:ColaDeEspera)
   Q.Desencolar(self);
   self.AbandoneElBanco
```

embebido dentro del método Encólese, como puede verse en la figura 6.b.

Si el cliente es atendido antes de que expire su tiempo máximo de permanencia en la cola, entonces la regla anterior debe ser descartada. Esta acción queda embebida dentro del método PaseElSiguiente de la cola de espera, como puede verse en la figura 6.c.

```
mut_meth PaseElSiguiente(Y : Cajero)
        ( not self.vacía )
        ifTrue C:= self.Desencolar;
                delRule ( conditions C.HoraEncolado +
                                C.TAguante = Reloj.HoraActual
                          actions  C.Impacientese(Q ) )
        ifFalse skip
```

Figura6.c.: método de la clase Cola de Espera

Análisis de las diferencias

• En NéOpus las reglas activas no forman parte de los métodos, sino que residen dentro de una base de reglas independiente. Con nuestro lenguaje es posible definir reglas en forma integrada dentro de los métodos. • En NéOpus existe una única regla general para todos los clientes, en nuestro modelo se crea una regla particular para cada cliente que ingresa en la cola de espera. Por lo tanto para evaluar si ha ocurrido el evento sólo serán analizados los clientes incluidos en la cola y no todos los clientes del banco. Por la misma razón tampoco resulta necesario colocar en la regla la condición de que C pertenezca a Q. • En NéOpus las reglas son persistentes, por lo tanto la regla TopeDeEspera continuará siendo evaluada aún cuando ningún cliente permanezca en la cola. En nuestro modelo la regla es efímera, luego de ejecutarse desaparece. Este comportamiento es adecuado ya que cuando el cliente abandona la cola ya no puede impacientarse. Si el mismo cliente decidiera regresar al banco y colocarse nuevamente en la cola de espera entonces el método Encólese generará una nueva regla que le permitirá impacientarse nuevamente.

7. Conclusiones

La programación basada en reglas es más adecuada que la programación orientada a objetos para la codificación de tareas reactivas. Existen propuestas que permiten a los programadores orientados a objetos usar ventajas de la programación basada en reglas dentro de un ambiente orientado a objetos, el resultado es un producto donde los dos paradigmas conviven (se integran) pero no se unifican

Hemos definido las bases para un lenguaje de programación orientado a objetos que permite expresar comportamiento activo en forma unificada. El comportamiento activo queda embebido dentro del comportamiento general de los objetos del sistema. Se respeta el encapsulamiento de los objetos y se aprovecha la jerarquía de clases para determinar el alcance de las reglas.

El comportamiento del sistema se va definiendo a lo largo de la ejecución, ya que cada objeto al desplegar su comportamiento (ejecutar un método) puede agregar y/o borrar mensajes activos del sistema. Es decir, cada objeto tiene la posibilidad de "legislar" (dictando y/o anulando reglas.) sobre el sistema.

El lenguaje permite expresar distintas formas de comportamiento activo: general, particular, persistente y efímero. Esto facilita el desarrollo de determinadas aplicaciones, por ejemplo, las reglas efímeras permiten modelizar eventos en forma natural, evitando de esta manera la necesidad de recurrir a un diseño artificioso (donde los eventos son tratados como objetos de primera clase, como en [Haythorn94]).

Por otra parte, la especificación formal provee una descripción precisa de las construcciones del lenguaje , facilitando el desarrollo de software confiable y dando las bases para testear su correctitud.

Bibliografía

[Atkinson87] Atkinson,R and J. Laursen , Opus: a Smalltalk production system, SigPlan Notices 22, 1987.

[Bertino94] E.Bertino, G.Guerrini and D Montesi, Deductive Object Databases, Proceedings ECOOP'94, July 1994, LNCS 821.

[Cardelli85] Cardelli, L.,Wegner P. On Understanding Types, Data Abstraction and Polymorphism. Computing Surveys, 17(4). 1985.

[Eick94] C. Eick and B. Czejdo, "Reactive Rules for C++ ", Journal Object Oriented Programming October 1993.

[Gatziu91] S. Gatziu, A. Geppert, K.Dittrich, "Integrating Active concepts into an object-oriented database system", Proceedings of the 3rd Workshop on database programming Languages, Nafplion 1991.

[Gehani91] N. Gehani, H.Jagadish, "Ode as an active database: constraints and reglas", Proceedings of the 17th International Conference on VLDB, Barcelona 1991.

[Haythorn94] Wayne Haythorn, "What is Object Oriented Design", Journal O O P April. 1994.

[Hennessy90] M.Hennessy, "The Semantics of Programming Languages: An elementary introduction using structural operational semantics" Copyright 1990 by J Wiley@Sons. England. Chapter six.

[Kappel94] G. Kappel, Rausch-Schott, W.Retschitzegger and S. Vieweg, "TriGS: Making a passive object-oriented database system active", Journal OO Programming, August 1994.

[Medeiros91] C. Medeiros and P.Pfeffer, "Object integrity using rules", Proceedings of ECOOP 1991, LNCS 512.

[Miranker93] Miranker, D, Burke, Steele, et al. "The C++ embeddable rule system", International Journal On Artificial Intelligence Tools, 2(1), 1993.

[Pachet95] F.Pachet, On the embeddability of production rules in object-oriented languages, Journal Object Oriented Programming, August 1995.

[Pons95] C.Pons, R.Giandini,G.Baum , "Modelos matemáticos para Objetos", Memorias de Panel'95 - XXI Conferencia Latinoamericana de Informática, Brasil, Julio 1995.

Aplicando modelos de procesos de *software* al desarrollo de aplicaciones hipermedia.

Jonás A. Montilva C.

Universidad de Los Andes
Facultad de Ingeniería
Escuela de Ingeniería de Sistemas
Departamento de Computación
Grupo de Investigación en Ingeniería de Datos y Conocimiento
Mérida, Venezuela
e-mail: jonas@ing.ula.ve

RESUMEN

La mayoría de los métodos y técnicas existentes para el desarrollo de aplicaciones multimedia e hipermedia se concentran en el diseño y construcción de hiperdocumentos, dejando de lado las otras fases del proceso de desarrollo. Otros ignoran conceptos fundamentales de la Ingeniería de *Software*, tales como abstracción, modularidad, re-utilización, validación y verificación. En este artículo se propone un método, basado en los modelos de procesos de la Ingeniería de *Software*, que cubre todo el ciclo de desarrollo de un hiperdocumento. El método emplea un modelo de referencia hipermedia orientado a objetos, el cual define formalmente la noción de hiperdocumento. El método aporta los siguientes beneficios: (1) proporciona una estructura amplia de división del trabajo que facilita la planificación del proyecto y la estimación de costos, tiempos y recursos; (2) agrega mayor visibilidad al proceso de desarrollo de hiperdocumentos; y (3) mejora la calidad de los hiperdocumentos producidos.

ABSTRACT

Most of the existing hypermedia and multimedia development methods and techniques concentrate on the design of hyperdocuments, ignoring the rest of the development phases. Others do not take into consideration the fundamental concepts of Software Engineering, such as data and functional abstraction, modularity, reuse, verification and validation. We propose in this paper a software engineering method for the development of hyperdocuments. The method covers all phases of the process. It is based on an object-oriented hypermedia reference model, which formalizes the notion of hyperdocument used by the method. The main benefits of the method are: (1) it helps to plan a hypermedia project by providing a comprehensive work breakdown structure that helps the project manager to estimate cost, time and resources; (2) it adds visibility to the hypermedia development process; and (3) it improves the quality of the hypermedia products.

1. INTRODUCCION

El desarrollo de aplicaciones hipermedia es un proceso complejo que ocasiona costos muy elevados, requiere gran inversión de tiempo y demanda de sus diseñadores una alta creatividad y el dominio de múltiples tecnologías audio-visuales. La complejidad del proceso está obviamente influenciada por los aspectos propios de la tecnología multimedia, tales como: el tipo de interacción usuario-sistema; el caracter dinámico de los principales tipos de datos multimedia (i.e., audio, video y animación); el caracter no-lineal de la organización del hiperdocumento; la navegación a través del espacio de información; y la naturaleza multidisciplinaria de los grupos de desarrollo de hiperdocumentos. El diseño de nuevos métodos para el desarrollo de sistemas o aplicaciones hipermedia es, por consiguiente, una actividad de investigación prioritaria e indispensable para el avance de esta nueva área tecnológica.

Durante los últimos cinco años han sido publicados en la literatura pertinente un número creciente de métodos y técnicas dedicadas a la producción de aplicaciones multimedia e hipermedia [1-6]. La mayoría de estos métodos se concentran en el diseño y construcción de la aplicación, ignorando las otras fases del proceso de desarrollo (p. ej., [3], [5] y [6]). Otros métodos cubren la mayoría o totalidad de las fases del proceso de desarrollo (p.ej., [1], [2] y [4]), pero hacen muy poco énfasis en aspectos fundamentales de la Ingeniería de *Software*, tales como el aseguramiento de la calidad, la gerencia del proyecto y la aplicación de los conceptos esenciales de esta disciplina: abstracción, modularidad, estructuración, re-utilización, validación y verificación, entre otros.

La Ingeniería de *Software* puede contribuir a mejorar significativamente el desarrollo de aplicaciones hipermedia, mediante la aplicación y adecuación de: (1) sus modelos de procesos y métodos de desarrollo; (2) sus técnicas para la especificación de requerimientos, el diseño, la verificación y la validación; y (3) los métodos de planificación y control del proyecto y el aseguramiento de la calidad. La principal contribución de ello se resume en el aumento de la calidad de las aplicaciones hipermedia y en un mejoramiento de la gestión del proyecto.

En este artículo se propone un método para el desarrollo de aplicaciones hipermedia (hiperdocumentos) basado en los modelos de procesos y conceptos de la Ingeniería de *Software*. Este método, a diferencia de otros tales como [3], [5] y [6], cubre todo el ciclo de desarrollo de una aplicación hipermedia y es el resultado de integrar tres modelos de procesos de *software*: el modelo de cascada, el modelo de prototipos y el modelo de re-utilización de componentes [7].

El diseño del método tomó en consideración las diferencias esenciales entre un programa y un hiperdocumento, las cuales se hacen evidentes a través de los modelos de referencia de la tecnología hipermedia [8, 9]. La definición de hiperdocumento empleada por el método se basa en un modelo de referencia hipermedia (HRM - *a Hypermedia Reference Model* [9]). Este

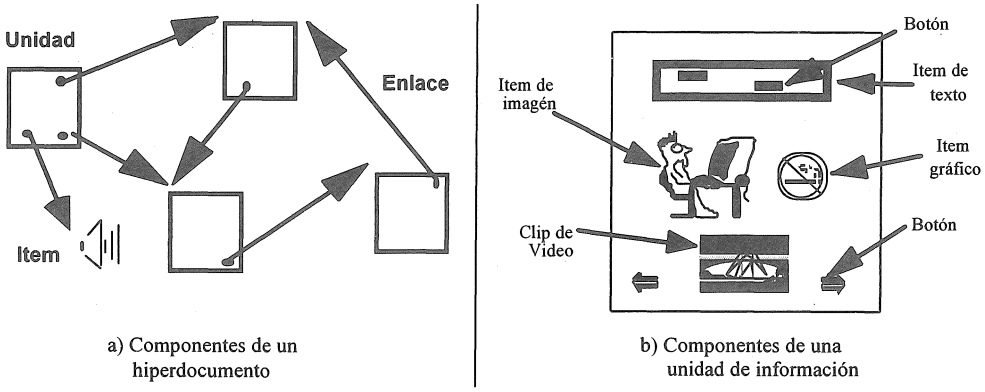
modelo define formalmente un hiperdocumento utilizando grafos dirigidos y describe sus componentes genéricos a través de un modelo de objetos.

El método es independiente de las herramientas existentes para el desarrollo de aplicaciones hipermedia. Emplea las técnicas de análisis y diseño orientadas a objetos, descritas en [11], las cuales permiten un modelado de la estructura y contenido del hiperdocumento mucho más natural, elegante y de facil comprensión que aquel que podría alcanzarse con los métodos imperativos convencionales.

El artículo está organizado como sigue. El modelo de referencia hipermedia (HRM) se describe en la sección 2. La sección 3 describe los objetivos y la estructura del método. Las fases, pasos y tareas del método se describen en la sección 4. Finalmente, en la sección 5 se discuten y los resultados y las conclusiones del trabajo. Los detalles del método, que por razones de espacio fueron omitidos en este artículo, se dan en [10].

2. HRM - UN MODELO DE REFERENCIA HIPERMEDIA

Las definiciones de hiperdocumento o aplicación hipermedia existentes en la literatura son muy variadas y se caracterizan por emplear una gran diversidad de términos, cuyas definiciones crean confusión debido a la ausencia de estándares. Para agregarle al método generalidad e independencia con respecto a las herramientas de desarrollo, se hizo necesario utilizar un modelo genérico o de referencia que describiese los conceptos fundamentales de la tecnología hipermedia. Se empleó un modelo propio, descrito en [9, 10], conocido como HRM (*Hypermedia Reference Model*). Este modelo emplea la orientación por objetos y se fundamenta en los conceptos de hypermedia utilizados por el conocido Modelo Dexter [8].



a) Componentes de un hiperdocumento

b) Componentes de una unidad de información

Figura 1. Estructura de un hiperdocumento

De acuerdo al modelo HRM, un hiperdocumento se define formalmente como un grafo dirigido *G(N, E)*, en el que *N* es un conjunto de *nodos de información* y *E* es un conjunto de arcos denominados *enlaces*, cada uno de los cuales conecta dos nodos de *N* (ver Fig. 1a). Los nodos de información pueden ser de dos tipos: unidad de información o ítem de información. Una *unidad de información* describe o refiere a un objeto (i.e., entidad, concepto o función) del dominio de la aplicación. Estructuralmente, una unidad es un objeto compuesto por un conjunto de ítems de información multimedia (p. ej., texto, gráficos, imágenes, pistas de audio o *clips* de video) y un conjunto de *botones* asociados a los enlaces (ver Fig.1.b).

Un enlace conecta una unidad fuente con otra unidad o ítem de destino. Un botón está asociado a un enlace. Al pulsar un botón se ocasiona la recuperación, en la base de datos, de la unidad o ítem de destino y su presentación o despliegue visual en pantalla. Un ítem de información es un objeto que contiene un dato de tipo multimedia. Estos datos se clasifican en estáticos (texto, gráficos e imágenes) y dinámicos (audio, video y animación). La Fig. 2 ilustra, mediante un modelo de objetos OMT, las clases de objetos que conforman un hiperdocumento y sus relaciones.

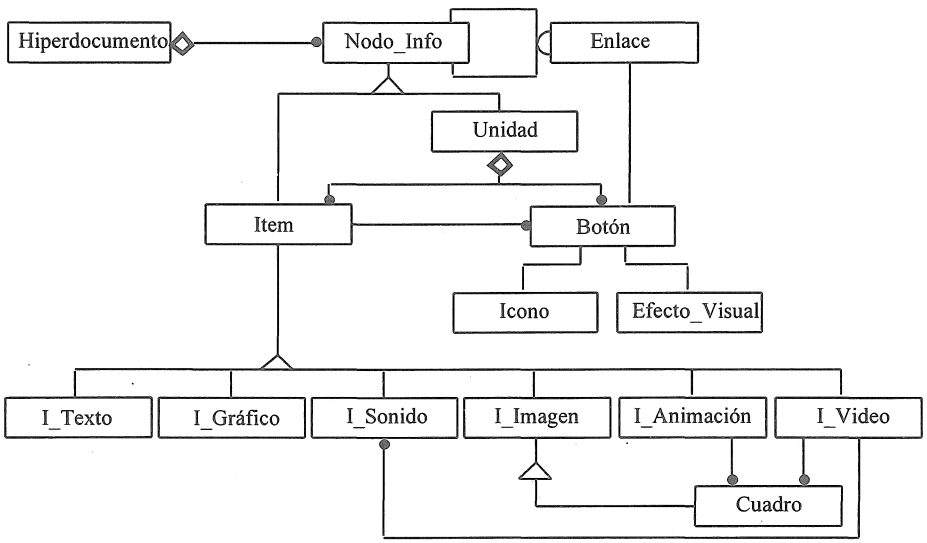


Figura 2. Modelo de objetos de un hiperdocumento

3. OBJETIVOS Y ESTRUCTURA DEL METODO

El método, aquí descrito, tiene por objetivos los siguientes:

1.- Guiar al grupo de profesionales que participan en el desarrollo de un hiperdocumento. El método debe responder a las preguntas siguientes: ¿ qué hacer? y ¿cómo hacerlo?

2.- Contribuir a la planificación del proyecto mediante la definición de las fases, pasos y tareas requeridas para desarrollar un hiperdocumento. Ello facilita la estimación de costos, tiempos y recursos requeridos.

3.- Asegurar la calidad del hiperdocumento producido.

El método se estructura en fases, tal como se ilustra en la Fig. 3. Cada fase se divide en pasos y estos, a su vez, en tareas. El proceso de desarrollo de un hiperdocumento se inicia con un análisis del dominio de aplicación del hiperdocumento (análisis del contexto); continua con la definición de los requerimientos, el diseño de su estructura y componentes; la producción textual, gráfica o audio-visual de sus componentes y la evaluación del hiperdocumento producido. El ciclo de desarrollo es evolutivo: se repite sucesivas veces, en base a la evaluación del hiperdocumento, hasta alcanzar una alta calidad del producto y un alto grado de satisfacción del usuario.

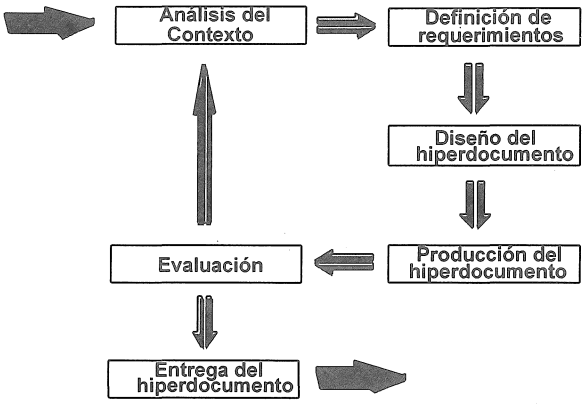


Figura 3: Fases del modelo de procesos

El método asume la existencia de tres actores: el cliente, el grupo de desarrollo y los usuarios. Se asume que existe una relación contractual entre el cliente y el grupo de desarrollo. El cliente contrata el desarrollo de un hiperdocumento a un grupo de expertos, quienes elaborán el producto para una comunidad no necesariamente conocida de usuarios.

4. FASES Y PASOS DEL METODO

Fase 1.- Análisis del contexto del hiperdocumento.- Los objetivos de esta fase son conocer las necesidades iniciales del cliente y familiarizarse con el dominio de la aplicación.

El proceso, descrito en la Fig. 4, se inicia con un análisis del dominio de aplicación o contexto en el que se usará el hiperdocumento. Durante este análisis se establece y describe someramente el dominio del problema. Se establece, también, una aproximación a las necesidades o requerimientos iniciales del cliente. Seguidamente, se describen los objetivos y el tema general o tópico del hiperdocumento y se realiza una investigación documental del tema escogido. En este paso, se puede elaborar un modelo preliminar de los elementos (i.e., objetos o procesos) que conforman el dominio de la aplicación; ello permite que el grupo de desarrollo obtenga rapidamente un mayor conocimiento del dominio de la aplicación hipermedia,. Finalmente, se establece el perfil del usuario potencial de la aplicación, en base a características tales como, su grado de instrucción, edad, nivel cultural, uso que le dará a la aplicación, etc.

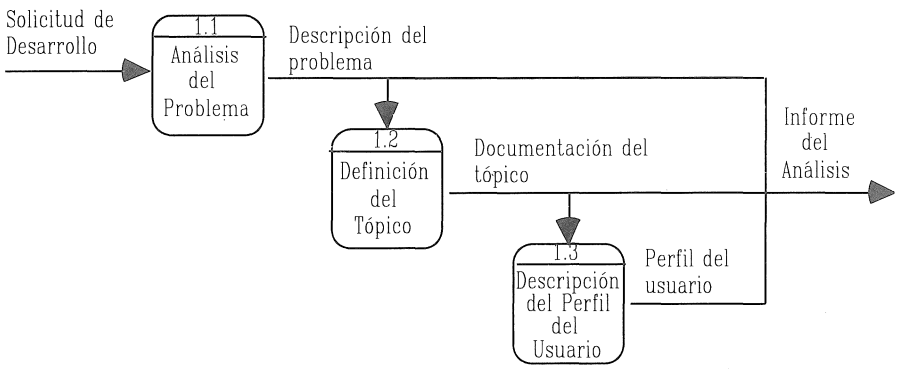


Figura 4. Fase de Análisis del Contexto del Hiperdocumento

Fase 2: Definición de requerimientos.- Los requerimientos que debe cumplir o satisfacer el hiperdocumento se definen y especifican en esta fase (Fig.5).

Se establecen primero los requerimientos funcionales de la aplicación hipermedia. Ello implica responder a la interrogante siguiente: ¿ que actividades realizará el usuario con el hiperdocumento ? Estas actividades se pueden representar mediante la construcción de un modelo funcional basado en los conocidos diagramas de flujo de datos empleados en OMT [11]. A continuación, se especifican los requerimientos de interacción, los cuales describen con detalle las características de la interfaz usuario-sistema, incluyendo los niveles de ayuda y orientación al usuario durante la navegación. Se continua con la especificación de las restricciones de desarrollo y operación del hiperdocumento; por ejemplo, los estándares de calidad exigidos, las herramientas de desarrollo disponibles o requeridas, el ambiente de operación (*hardware* bajo el cual operará el hiperdocumento) y otros requerimientos que restringan o delimiten el proceso de desarrollo y la

operación del hiperdocumento. Finalmente, se establecen los requerimientos de calidad del producto hipermedia; por ejemplo, su eficiencia (tiempos de respuesta y volumen de almacenamiento de datos multimedia requeridos); la re-utilización de sus componentes, su portabilidad; utilidad y facilidad de uso y su interoperabilidad con otros hiperdocumentos o bases de datos multimedia. Los criterios de diseño del hiperdocumento se establecen, también, en este último paso. Estos criterios incluyen, entre otros, los siguientes: riqueza del contenido de información, facilidad de acceso a la información, consistencia entre los elementos de la aplicación, predictibilidad (anticipación al resultado de una operación) y legibilidad [12].

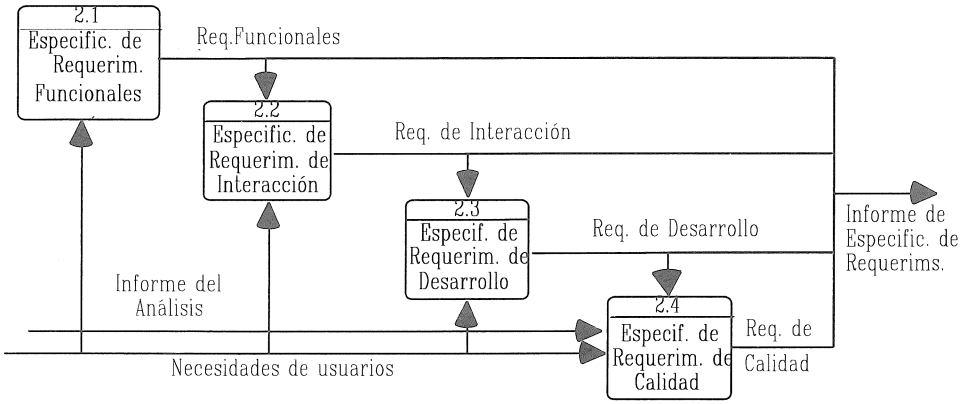


Figura 5. Fase de especificación de requerimientos

Fase 3: Diseño del hiperdocumento.- En esta fase se diseña: (1) la estructura del hiperdocumento; (2) cada una de las unidades de información que componen dicha estructura; y (3) los ítems que conforman cada una de sus unidades (Fig. 6).

Previo al inicio de esta fase se deben seleccionar las herramientas de desarrollo, esto es, el conjunto de programas o paquetes multimedia que se utilizarán para construir el hiperdocumento. La fase culmina con el desarrollo de un prototipo que ilustra la estructura del hiperdocumento y su interacción con el usuario. Este prototipo es utilizado, por un grupo representativo de usuarios, para validar el diseño del hiperdocumento. Finalmente, las especificaciones de diseño se verifican con respecto a las especificaciones de requerimientos establecidas en la Fase 2.

El *diseño de la estructura* consiste en las siguientes tareas: (a) la descomposición del tópico del hiperdocumento en sub-tópicos o secciones temáticas; (b) la elaboración de un modelo conceptual para cada sección temática; y (3) el diseño de la estructura interna de cada una de estas secciones temáticas. Para alcanzar un alto grado de mantenibilidad y comprensibilidad del hiperdocumento, su estructura temática debe ser modular y basada en los criterios de cohesión y

acoplamiento; ello implica la busqueda de una alta cohesión en cada sección y un bajo acoplamiento entre secciones (Fig. 7.a). La ventaja adicional que se deriva de aplicar este criterio es que permite que las secciones puedan ser re-utilizadas en otras aplicaciones similares.

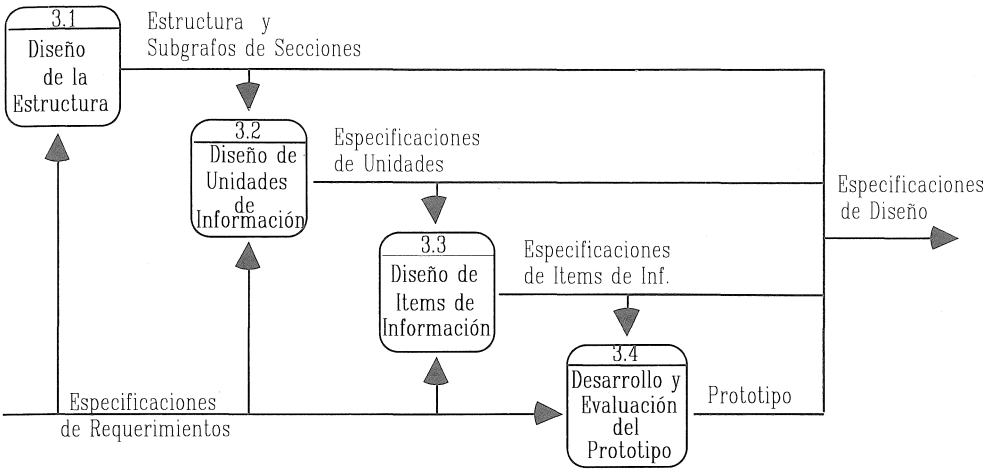
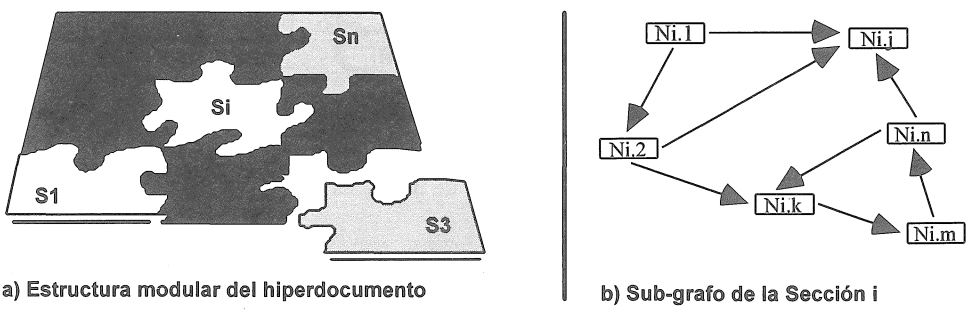


Figura 6. Fase de diseño del hiperdocumento

El modelado conceptual de cada sección temática consiste en representar las entidades del dominio de la aplicación enmarcadas por esa sección. El modelado de objetos OMT [11], la técnica descrita en [5] o los diagramas RMDM [4] bien pueden ser empleados para este propósito; aunque es, también, posible que los requerimientos impongan una descomposición funcional del dominio de aplicación (p. ej., en hiperdocumentos que describen procesos o procedimientos), para lo cual el modelado funcional basado en diagramas de flujo de datos podría ser de mayor utilidad.



a) Estructura modular del hiperdocumento

b) Sub-grafo de la Sección i

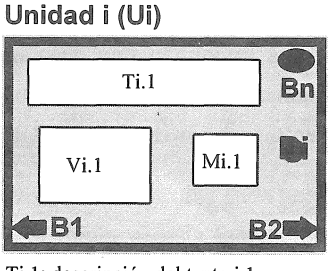
Figura 7. Diseño de la estructura del hiperdocumento

El diseño de la estructura interna de cada sección se elabora en base a su modelo conceptual elaborado en la tarea anterior. Se produce un sub-grafo para cada sección que muestra la

estructura temática inicial de la sección organizada en nodos de información interconectados (Fig. 7.b). Cada nodo corresponde a una clase de objetos (o función) del modelo conceptual correspondiente a la sección. Las conexiones entre seçciones deben, también, establecerse en este paso.

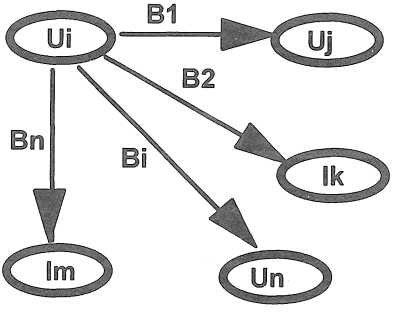
El *diseño de unidades* consiste en la especificación del contenido de cada unidad. Ello implica: (1) definir los ítems de información que cada unidad contendrá y los tipos de medios (texto, audio, video, etc.) que se emplearán para presentar estos ítems; (2) delinear o describir el contenido de cada ítem, así como establecer su ubicación en la unidad; y (3) definir los enlaces de cada unidad y sus botones correspondientes (Fig. 8.a). El diseño de cada sección temática culmina con la elaboración de un modelo dinámico para cada unidad, el cual define los cambios de estado de la unidad ocasionado por la activación de sus diferentes botones (Fig. 8.b). Estos modelos dinámicos definen la navegación a través del espacio de información del hiper-documento.

El *diseño de ítems* consiste en especificar la estética o los detalles artísticos del contenido de cada uno de los ítems que integran cada unidad; ello incluye el diseño de textos, gráficos, imágenes, pistas de sonido, *clips* de video y animación. Así, por ejemplo, para una pista de audio musical se deberá especificar la calidad del audio (8 o 18 bits; estéreo o monofónico), el tipo de melodía, su duración, etc.



Ti.1: descripción del texto i.1
Vi.1: descripción del video i.1
Mi.1: descripción de imagen i.1

a) Diseño de una unidad



b) Modelo dinámico de una unidad

Figura 8. Diseño de unidades

Fase 4: Producción del hiperdocumento.- En esta fase se procede a la implementación del diseño usando las herramientas de desarrollo seleccionadas al inicio de la fase 3. La fase se divide en los pasos señalados en la Fig.9.

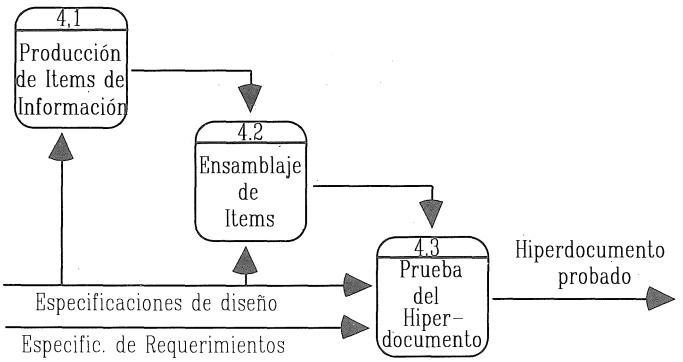


Figura 9. Fase de producción del hiperdocumento

La produccion de los ítems de información involucra la captura, digitalización, edición y almacenamiento de los ítems de audio, fotografía, video y animación; así como la transcripción de los ítems de texto y el dibujo de los ítems gráficos que componen las diferentes unidades de información del hiperdocumento. Producir ítems de audio, video y animación es una labor que demanda un alto grado de creatividad y el dominio de tecnologías audio-visuales. Una vez finalizada la producción de ítems, estos deben ensamblarse en el prototipo desarrollado en la fase 3. El hiperdocumento resultante se somete a un proceso de verificación con respecto a los requerimientos especificados en la fase 2, a fin de determinar su calidad y el cumplimento de tales requerimientos.

Fase 5: Evaluación y documentación del hiperdocumento.- La Fig. 10 ilustra la fase final del método.

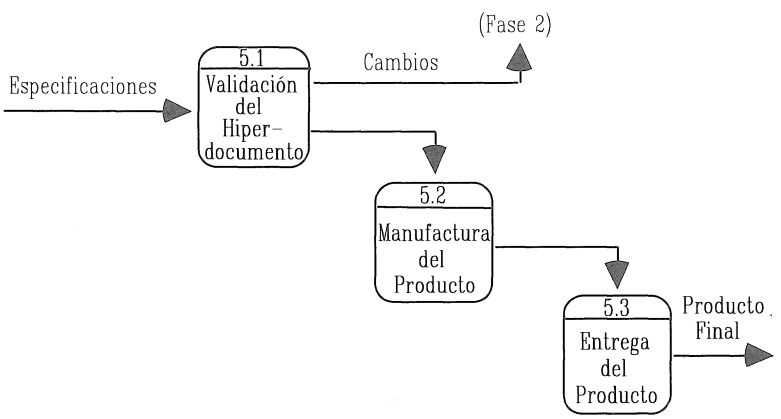


Figura 10. Fase de evaluación y entrega del producto

Esta fase consiste en: (1) la validación y corrección final del hiperdocumento con el objetivo de asegurarse que éste satisface los requerimientos impuestos por el cliente; (2) la elaboración de

la documentación que introduce y describe cómo se utiliza el hiperdocumento; la manufactura o grabación del hiperdocumento en los medios exigidos por el cliente (ej., discos ópticos, discos compactos o disquetes); y (3) la entrega del producto final al cliente.

La validación puede implicar nuevos cambios al hiperdocumento que podrían dar origen a la repetición o iteración del proceso a partir de la fase 2. El ciclo finaliza cuando se ha logrado producir un hiperdocumento de alta calidad; esto es, un hiperdocumento que cumple con los requerimientos, los atributos de calidad y los criterios de diseño impuestos por el cliente.

5. RESULTADOS Y CONCLUSIONES

El método descrito en este artículo fue motivado por la idea de que un uso más riguroso de los métodos y técnicas de la Ingeniería de *Software,* en el desarrollo de aplicaciones hipermedia, puede contribuir a mejorar significativamente la calidad de los hiperdocumentos e incrementar el grado de satisfacción de sus usuarios.

La contribución del método al desarrollo de aplicaciones hipermedia se resume en: (1) ayuda al gerente del proyecto a lograr una mejor división del trabajo requerido en el proceso de desarrollo de hiperdocumentos y, por ende, a lograr una mejor planificación del proyecto y mejores estimaciones de costos y tiempos; (2) establece una mayor visibilidad y claridad al complicado proceso de desarrollo; y (3) produce hiperdocumentos de mejor calidad. La calidad es medida, en el método, en función de los grados de satisfacción del usuario, estructuración, re-utilización, mantenibilidad, comprensibilidad y modularidad.

El método propuesto difiere de los métodos citados [1-6] en varios aspectos. El nuestro cubre todas las fases del ciclo de desarrollo de un hiperdocumento (desde el análisis del dominio hasta la entrega del producto), a diferencia del método HDM de Garzotto, Paolini and Schwabe [3] y la técnica HMT de Nemetz, Valdeni y Cardoso [5], las cuales cubren sólo la fase de diseño. El método OOHDM [6] se asemeja al nuestro en el uso de la orientación a objetos, pero deja de lado la fase de evaluación y entrega del producto; fase fundamental para determinar la calidad del producto. Nuestro método hace mayor énfasis en la calidad del hiperdocumento que los métodos de Bunzel [1], Martin [2] y Isakowitz, et al [4] y proporciona una estructuración de fases, pasos y tareas mucho más clara, elegante y orientada a la planificación del proyecto. Cabe destacar, sin embargo, que el método de Bunzel es único en cuanto al diseño de multimedios se refiere; pues considera en detalle las actividades de diseño y producción de imágenes, pistas de audio y video.

El método es probado actualmente en la elaboración de hiperdocumentos educativos y en la creación de hipermapas para sistemas de información geográfica [13].

Referencias

[1] Bunzel, M.J. *Multimedia Applications Development Using DVI Technology*. McGraw Hill, 1990.

[2] Martin, J. *Hyperdocuments and how to create them*. Prentice-Hall, 1990.

[3] Garzotto, F., Paolini, P. and Schwabe, D. HDM - A Model Based Approach to Hypermedia Application Design. *ACM Transactions on Office Information System*, Vol.11, No.1, 1993, p.1-26.

[4] Isakowitz, T., Stohr, E.A., Balasubramanian, P. RMM: A Methodology for Structured Hypermedia Design. *Comm. of the ACM.*, Vol. 38, No. 8, August, 1995, p. 34-44.

[5] Nemetz, F., Valdeni, J., Cardoso, A. Una Técnica para Modelagem de Aplicações Hipermídia e um Exemplo de Uso (en portugués). *Actas de la XXI Conferencia Latino-Americana de Informática (PANEL'95)*, Canela, Brazil, 1995, pp.261-274.

[6] Schwabe, D. and Rossi, G. The Object-Oriented Hypermedia Design Model. *Comm. of the ACM.*, Vol. 38, No. 8, August, 1995, p. 45-46.

[7] Sommerville, I. *Software Engineering*. 4th. Edition. Addison-Wesley, Reading, MA. 1992.

[8] Halasz, F. The Dexter Hypertext Reference Model. *Comm.of the ACM*, Vol.37, No.2, 1994.

[9] Montilva, J.A. *An Integration Method Applied to the Design of a Data/Knowledge Model for Multimedia and Spatial Applications*. Ph.D. Thesis. University of Leeds, School of Computing Studies, Leeds, UK. March, 1993.

[10] Montilva, J. A. *Métodos y técnicas para el desarrollo de aplicaciones multimedia*. Informe técnico en preparación. Universidad de Los Andes. Facultad de Ingeniería. Departamento de Computación, Grupo GIDyC. 1996.

[11] Rumbaugh, J. et al. *Object-Oriented Modeling and Design*. Prentice Hall International. 1991.

[12] Garzotto, F., Mainetti, L. and Paolini, P. Hypermedia Design Analysis, and Evaluation Issues. *Comm. of the ACM.*, Vol. 38, No. 8, August, 1995, p. 74-86.

[13] Montilva, J. A. hypermap data model for integrating multimedia and geographical information systems. *Actas de la XXI Conferencia Latino-Americana de Informática (PANEL'95)*. Canela, Brazil, Julio, 1995, pp. 1139-1150.

Diseño y Consultas de Aplicaciones Hipermedias

Silvia Gordillo

Comisión de Investigaciones Científicas de la Pcia. de Bs. As, y LIFIA,
Universidad Nacional de La Plata.
Argentina
gordillo@info.unlp.edu.ar

Alicia Díaz

Consejo Nacional de Investigaciones Científicas y Técnicas, y LIFIA,
Universidad Nacional de La Plata.
Argentina
alicia@info.unlp.edu.ar

Abstract

En este artículo proponemos un modelo Orientado a Objetos para el diseño de aplicaciones hipermedia que toma las ventajas provistas por el paradigma tales como tipos complejos, definidos por el usuario, atributos no atómicos y no convencionales, etc. De esta forma permitimos no solamente la modelización de la información sino también, formas alternativas para acceder a la misma.

También se presenta un lenguaje de consultas que combina características de los lenguajes de bases de datos orientadas a objetos, con las primitivas de navegación de los hipermedias. El lenguaje definido ofrece tanto la posibilidad de realizar consultas a la información almacenada, como a la información del esquema de la aplicación.

PEDECIBA INFORMATICA

1. Introducción

La tecnología hipermedia a emergido como una excelente alternativa para la construcción de sistemas de información complejos. Esta tecnología ha sido recientemente aplicada en diferentes dominios tales como educación, museos virtuales, bases de datos heterogéneas, ingeniería de software, etc.

Sin embargo, esta tecnología por ser algo reciente, presenta algunos problemas aún no resueltos. Uno de estos problemas se basa en la naturaleza de los hipermedia, y está directamente relacionado con su estructura subyacente de *nodos* y *links,* la cual permite la navegación natural de la información. Si bien esto puede resultar una ventaja en muchos aspectos, también es cierto que conocidos conceptos de la ingeniería de software, como calidad, reusabilidad, modificabilidad no pueden ser aplicados. Por otra parte, la navegación como principal método de acceso, resulta eficiente cuando el usuario tiene un conocimiento general de la información , pero puede causar algunos inconveniente si se desea acceder a información específica, e incluso puede causar desorientación, sobre todo cuando se esta trabajando en un gran hiperespacio.

En este sentido, existen algunas propuestas en las que se definen modelos para el diseño de hipermedias, como son [6], [7], [13], [15], que se describen metodologías de diseño de aplicaciones hipermedias, pero no tratan específicamente el tema de las consultas a estas aplicaciones.

Existen también, algunos enfoques que tratan de agregar las capacidades de consultas a aplicaciones hipermedias como una posibilidad adicional al acceso navegacional. Algunos de estos enfoques usan las bases de datos relacionales como una forma de proveer conocimiento sobre la estructura del grafo y los tipos de nodos. Por ejemplo, [16] combina un sistema de hipermedia con un DBMS, y de esta manera representa la información a través de atributos atómicos que pueden ser consultados. [12] propone una base de datos Orientada a Objetos como soporte para la aplicación hipermedia, con lo cual la información es accedida usando el lenguaje de consulta de la base de datos.

En cambio el objetivo del presente trabajo es desarrollar un modelo que agregue los conceptos mencionados en el párrafo anterior estructurando la información de manera de obtener un esquema de la aplicación que sirva como base, tanto para el diseñador contando con herramientas que le permitan un cierto nivel de comprensión de la estructura de la aplicación; como para el usuario permitiéndole el acceso a la información no solamente a través de la

navegación, sino también por medio de un lenguaje de consultas. La ventaja del modelo es que pone el énfasis en conservar la flexibilidad propia de este campo de manera de preservar las facilidades que provee, como por ejemplo manejar distintas perspectivas de la información, tener una interfaz multimedial, facilidades de navegación, etc.

En la sección 2 describimos las principales características del modelo y los diferentes niveles de diseño para construir aplicaciones hipermedias. En la Sección 3 se describe el lenguaje de consultas definido y en la sección 4 se discuten las principales ventajas del modelo.

2. Un modelo OO para diseñar aplicaciones hipermedias

El modelo propuesto se basa en el paradigma orientado a objetos ya que permite definir un modelo donde datos no-convencionales, como los usados en hipermedias, pueden ser representados en forma natural e implementarse fácilmente las ventajas mencionadas anteriormente [8], [9], [11], [4]. Consiste básicamente de dos niveles de diseño, el primero captura el dominio de la aplicación en un alto nivel de abstracción, donde se definen globalmente las entidades y relaciones; en el segundo nivel de diseño, se definen las perspectivas de los datos, los de navegación y las interfaz. En la sección 2.1 y 2.2 describimos en detalle cada uno de estos niveles.

2.1 *Diseño de alto nivel*

El objetivo de este paso es diseñar la aplicación hipermedia en una forma abstracta dejando de lado los detalles y concentrándose solamente en los elementos más importantes y en sus relaciones. Este mecanismo es similar al proceso de diseño propuesto en [14], el cual distingue las clases de entidades de la aplicación, organizadas en jerarquías de especialización y composición, y las relaciones conceptuales que existen entre ellas. En este nivel solo se describen los objetos de la aplicación y las relaciones, junto con sus propiedades y comportamiento, en cambio no se modela ninguna información acerca de las características de navegación, objetos de interfaz y detalles de implementación.

2.1.1 Entidades

Cada tipo de entidad en el dominio de la aplicación es modelado como una clase de nodo, y se define por un conjunto de propiedades y comportamiento. Estas clases de nodo se relacionan a través de las relaciones de subclase (*es_un*) y de composición (*es_parte_de*), definiendo dos jerarquías de clases independientes.

Un atributo (propiedad) se define por su nombre y su dominio. Los atributos son clasificados, de acuerdo a su dominio, en siete categorías: a*tributo atómico*: su dominio es una clase primitiva (en nuestro modelo texto, sonido, imagen, video, etc. son clases primitivas); a*tributo complejo*: toman valor en clases definidas por el usuario; a*tributo compuesto*: su dominio es una clase no-primitiva y soporta la semántica de la relación *es_parte_de; atributo simple*: es instanciado con un solo valor del dominio; *atributo multivaluado*: es definido a través del los constructores *set_of* o *list_of; atributo multidominio*: puede tomar valor en varios dominios; a*tributo derivado*: toma valor en tiempo de ejecución a través de la evaluación de una función.

Los atributos multidominio se utilizan para implementar diferentes perspectivas de la misma información, ya que permiten visualizar la información a través de diferentes medios.

Existe una clase llamada *Nodo* que es la raíz de la jerarquía de clases de nodos y que soporta la relación *es-un*.

2.1.2 Relaciones

Las relaciones modelan las relaciones conceptuales entre dos o más entidades de la aplicación. Desde el punto de vista de una aplicación hipermedia, modelar las relaciones sirve para permitir al usuario navegar naturalmente a través de la información.

En este modelo las relaciones son objetos y por lo tanto se definen a través de clases que son organizadas en una jerarquía de especialización/generalización, donde la clase raíz se llama *Link*.

La clase *Link* define algunos atributos que sirven para especificar las clases de nodos fuente (atributo *From*) y destino (atributo *To*), cardinalidad y comportamiento para la navegación. Una clase de relación hereda de la clase *Link* y probablemente agregue sus propios atributos.

Es importante destacar que el modelo aquí propuesto distingue entre dos tipos de relaciones navegables, aquellas que describen relaciones particulares entre objetos y la relación *es_parte_de*, la cual determina composición. La composición es un caso especial, ya que define una semántica de navegación particular, permitiendo navegar un

objeto compuesto como una unidad o cada una de sus componentes a partir de su raíz. Este tipo de relación es tratada en otros modelos de hipermedia como [3].

2.2 *Diseño de bajo nivel*

En este nivel el objetivo es:

- diseñar la estructura navegacional definiendo los anchors
- definir perspectivas para visualizar la información, y
- describir los detalles para la interfaz.

Los anchors son los que permiten definir los distintos caminos de navegación que existen a partir de un objeto particular.

Una de las características que, como mencionamos anteriormente debe conservarse cuando se define un modelo para aplicaciones hipermedia, es la de poder definir distintas perspectivas para la misma información. Estas distintas perspectivas de definen por medio del concepto de *Ejemplar* [10]. Cada clase de nodo tiene asociado al menos un ejemplar.

2.2.1 **Ejemplar**

Un ejemplar se define por una lista de atributos que son seleccionados desde aquellos existentes en la clase de nodo asociada; mas el comportamiento esperado. Además, estos tienen los anchors correspondientes a los links que se visualizarán en la perspectiva que implemente este ejemplar.

Cuando se construye un ejemplar se elige sólo un dominio para aquellos atributos multidominio, es decir, se decide cuál va a ser el dominio del atributo para ese ejemplar.

De la misma manera que las clases de nodos y links, los ejemplares se organizan en una jerarquía de especialización/generalización donde la clase raíz se llama *Ejemplar*.

Los ejemplares podrían agregar nuevas relaciones para modelar relaciones excepcionales entre ejemplares. Estas clases de relaciones se modelan vía clases de links, pero donde el dominio de los atributos to y from son clases de ejemplares. Esencialmente, esta facilidad permite navegar las distintas perspectivas de la misma información.

Los dos niveles de diseño previamente descriptos definen el esquema del modelo, es decir el esquema se compone de tres jerarquías: clases de nodos describiendo las entidades de la aplicación, clases de links para las relaciones y clases de ejemplares, las que describen las diferentes perspectivas y los anchors. Este esquema es la base para el lenguaje de consulta que se presentará en la próxima sección.

Para ejemplificar nuestro modelo nos basaremos en la aplicación "*The Way Things Work of DK*" (*). esta es una aplicación educativa donde el usuario puede navegar para obtener aspectos tales como: cómo funciona una máquina, sus principios básicos asociados, la historia del invento hasta la biografía de su autor. La figura 1 muestra el esquema de esta aplicación, donde se especifican algunas de las entidades y relaciones encontradas en ella. Las relaciones descriptas a través de un diamante identifican la relación *es_parte_de*.

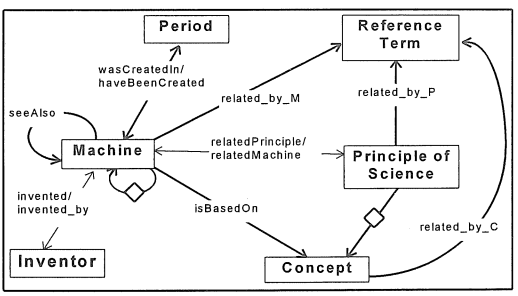


Figura 1: diseño de E-R de la aplicación The Way Things Work.

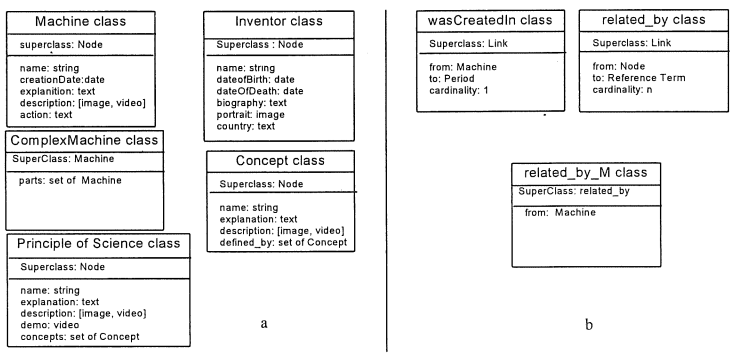


Figura 2: a.Ejemplos de clases Nodos b.Ejemplos de clases Link

En la figura 2 mostramos las clases de nodos y links para el ejemplo anterior, donde el lector puede ver sus atributos. En particular la Figura 2.a permite observar las clases de nodos, donde la clase de nodo *ComplexMachine* es subclase de la clase *Machine*.

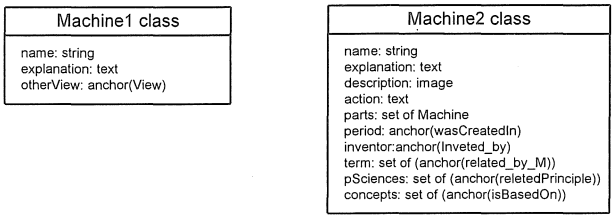


Figura 3: Dos ejemplares para la clase de nodo Machine.

La figura 3 muestra dos perspectivas posibles para el nodo Machine, la primera con el nombre e información general, mas un anchor al dibujo de la máquina y la segunda con información detallada mas la imagen de la misma.

3. Consultas a hipertextos

Con las bases de las definiciones previas, y a partir de la construcción del esquema , es posible definir un lenguaje de consultas del estilo de los lenguajes de consultas de las bases de datos pero además con la capacidad de navegación mas características propias de las aplicaciones hipermedias.

Los usuarios pueden acceder a la información vía dos formas: modo navegación o modo consulta. La primera es la manera tradicional de acceder a las aplicaciones hipermedias, en el segundo modo, en cambio, es posible acceder tanto a la información contenida en la aplicación como también a la información del esquema consultando alguna de las tres jerarquías de clases (*Nodos*, *Link* y *Ejemplar*).

El modelo de consultas está basado en el modelo definido por [8], donde se han realizado algunos cambios necesarios para considerar las características propias de las aplicaciones hipermedias.

3.1 *Consultando la Información*

Este tipo de consultas son orientadas al usuario, ya que estas permiten recuperar la información almacenada, donde el operador más importante es la *selección de objetos*. También existen otros constructores para manejar objetos compuestos, relaciones y propiedades (a través del operador *project*)

3.1.1 Operación de Selección

La selección de objetos desde una o más clases, cuando una condición es verdadera, es similar a la selección tradicional de las bases de datos relacionales.

***select** [ListaClases] **from** AsocVarObj **in** [fuente] [**where** condición]*

donde:

ListaClases: es la especificación de las clases resultantes.

AsocVarOobj: indica la asociación de las variables a los conjuntos de instancias.

fuente: es el hipertexto o el conjunto de objetos a ser consultados.

condición: es una combinación de predicados booleanos..

[...]: indica parámetros opcionales.

El resultado de esta consulta es un hipertexto formado por las clases que aparecen en la lista de clases resultantes más la relaciones que existen entre ellas (en otras palabras el resultado es un subhipertexto creado, a partir del hipertexto original, con solamente aquellas clases que aparecen involucradas en la consulta junto con las relaciones entre ellas).

El siguiente ejemplo recupera los inventores nacidos antes de 1900 y las máquinas que inventaron.

*Q := **select** Machine, Inventor **from** Inventor :i, Machine :m **where** (i.dateofBirth < 1900)*

3.1.1.1 *Cuantificadores*

La condición puede incluir atributos multivaluados, por esta razón el modelo incluye dos cuantificadores: EACH y EXIST, que corresponden a los cuantificadores universal y existencial respectivamente.

El siguiente ejemplo recupera maquinas que usan un motor eléctrico.

*Q := **Select** Machine **from** Machine :m **where exits** m.parts = "electrical engine"*

3.1.1.2 *Operadores sobre Conjuntos*

El predicado de selección puede involucrar atributos multivaluados. Para describirlos usamos los operadores SET y LIST [8], los que nos permiten trabajar con conjuntos y listas de atributos.

3.1.2 Operación de Proyección

La operación de proyección permite seleccionar algunos atributos de objetos pertenecientes a una clase. El formato de la operación es:

select *[ListaClases]* ***from*** *AsocVarObj* ***in*** *[fuente] [**where** condición]*

donde la definición de *ListaClases, AsocVarObj, fuente y condición* es la misma que en la operación de selección.

El resultado de esta consulta es siempre un conjunto de listas con los atributos seleccionados.

En el siguiente ejemplo, el resultado es una lista con los nombres de inventores nacidos antes de 1900.

Q := select *Inventor.name* ***from*** *Inventor :i* ***where*** *(i.dateofBirth < 1900)*

3.1.3 Operación Related_By

Esta operación permite establecer si existe una relación entre dos objetos nodos. El resultado de la operación es un valor booleano que es verdadero cuando la relación existe y falso en caso contrario.

La sintaxis de esta consulta es:

related_by *(ObjetoFuente, NombreRelación, ObjetoDestino)*

El ejemplo permite recuperar los inventores de aquellas máquinas creadas durante la revolución industrial.

Q := select *Inventor, Machine* ***from*** *Machine:m, Period :p* ***where*** *(p.name= "Industrial Revolution" and* ***related_by*** *(m, WasCreatedIn, p))*

3.1.3.1 *Operación de Asignación*

A través de esta operación se permite almacenar el resultado de las consultas. Esta operación es la base que nos permite realizar nuevas consultas sobre el resultado obtenido de una previa.

3.2 *Consultando el esquema*

Este tipo de consultas, si bien pueden ser útiles para el usuario, están especialmente formuladas para asistir al diseñador de una aplicación, de manera por ejemplo, que pueda obtener información sobre la jerarquías del esquema del hipertexto.

3.2.1 Operaciónes sobre jerarquías

La operación ***Hierarchy*** es usada para establecer cuales son las superclases o subclases de una clase dada.

hierarchy*↑** *NombreClase* ***From *Jerarquía*

or

hierarchy*↓** *NombreClase* ***From *Jerarquía*

donde:

NombreClase: es el nombre de una clase de alguna de las jerarquías del esquema.

Jerarquía: es el nombre de algunas de las jerarquías *Nodo*, *Link* o *Ejemplar*

La diferencia entre **hierarchy↑** y **hierarchy↓** es que la primera recupera nombres de superclases, en cambio la segunda nombres de subclases.

3.2.2 Operaciones sobre clases

La operación ***properties*** recupera las propiedades definidas en una clase.

properties *NombreCklase* ***from*** *Jerarquía*

NombreClase y Jerarquía se definen de la misma manera que para la operación **Hierarchy**.

Las operaciones ***source*** y ***traget*** recuperan, respectivamente, las clases de nodos origen y el destino de una relación particular.

source *NombreClaseLink*

target *NombreClaseLink*

El operador **exemplar** permite conocer los nombres de los ejemplares asociados a una clase de nodo particular.

exemplar *NombreClaseNodo*

Existen dos operaciones ***related_to*** y ***related_from*** que permiten determinar respectivamente, qué clases de nodos son alcanzables desde una clase específica y todas las clases de nodos desde las cuales es alcanzable una clases particular.

related_to *NombreClaseNodo*

related_from *NombreClaseNodo*

4. Conclusiones

Hemos presentado un modelo para aplicaciones hipermedias, el cual provee las bases para estructurar información a partir de los conceptos de Nodos, Relaciones y Ejemplares. Usando este modelo se puede diseñar una aplicación en diferentes niveles de abstracción, de la misma forma que ocurre en los métodos de ingeniería de software modernos, sin perder la facilidades tradicionales de los hipermedias. El mantenimiento de las aplicaciones diseñadas con este modelo resulta mucho mas simple que cuando se usa el enfoque tradicional porque se permite un mejor entendimiento del dominio de la aplicación.

La posibilidad de definir un esquema y un lenguaje de consultas incrementa las características para manipular aplicaciones hipermediales y provee herramientas adicionales para recuperar información, eliminando de alguna manera la desorientación de los usuarios. Esta capacidad también es muy útil para los diseñadores ya que pueden consultar el esquema aumentando su conocimiento sobre el dominio de la aplicación.

(*) The Way Things Work es trademark de Dorling Kindersley MULTIMEDIA.

5. Referencias

[1] B. Amann, M. Scholl. *"Gram: A graph data model and query language"*.Proceedings of the 4th ACM Conference on Hypertext and Hypermedia (ECHT'92), Milano, Italy, December 1992

[2] B. Amann, M. Scholl. *"Quering typed hypertexts in Multicard/O2"*. Proceedings of the 5th ACM Conference on Hypertext and Hypermedia (ECHT'94), Edinburgh, September, 1994.

[3] V. Chistiphides, A. Rizk. *"Quering structured documents with hypertext links using OODBMS"*. Proceedings of the 5th ACM Conference on Hypertext and Hypermedia (ECHT'94), Edinburgh, September, 1994.

[4] D. Chorafas and H. Steinmann. "*Object Oriented Databases*". PTR Prentice Hall, 1993

[5] M.P. Concens, A. Mendelzon. *"GaphLog: a visual formalism for real life recursion*". Proceedings of the ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, pp 404-416, Nashville, Tennessee, 1990.

[6] F. Garzotto, P. Paolini, D. Shwabe, *"HDM - A Model Based Approach to Hypermedia Applications Design"*, ACM Trans. Info. Syst. 11, 1 (Jan.1993) pp 1-26.

[7] T. Isakowitz, E. A. Stohr, P. Balasubramaninan, *"RMM: A Methodology for Structured Hypermedia Design"*, Communications of the ACM, August 1995.

[8] W. Kim. *"Introduction to Object Oriented Databases"*, MIT Press, 1990.

[9] W. Kim, *"Modern Database Systems"*,. ACM Press, 1995.

[10] W. LaLonde, *"Designing Families of Data Types Using Exemplars"*, ACM Toplas, April 1989.

[11] M. Loomis. *"Object Oriented Databases. The Essentials"*. Addison Wesley Publishing Company, 1994

[12] M. Marmann and G. Schlargeter. *"Towards a Better Support for Hypermedia Authoring: The HYDESIGN Model"*. Proceedings of the 4th ACM Conference on Hypertext and Hypermedia (ECHT'92), Milano, Italy, December 1992

[13] J. Nanard, M. Nanard. "*Using structured types to incorporate knowledge in hypertext*". Proceedings of Hypertext'91, pages 329-343, December 1991.

[14] J.J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen. *"Object Oriented Modeling and Design"*. Prentice Hall Inc., 1991.

[15] D. Schwabe, G. Rossi, S. Barbosa. *"Systematic Hypermedia Design with OOHDM"*. Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences, Maui, Hawaii (Jan. 1995).

[16] H. Van Dyke Parunak. *:Don't link me in: Set based hypermedia for taxonomic reasoning"*. Proceedings of Hypertext'91, pages 233-242, December 1991.

HSM - MODELAGEM SEMÂNTICA PARA APLICAÇAO HIPERTEXTO

Marcelo Ladeira[†], Joice Lee Otsuka, José Vadeni de Lima, Ricardo Azambuja Silveira

Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)

Caixa Postal 15064 - CEP 91501-970 Porto Alegre, RS - BRASIL

Email: {mladeira, otsuka,valdeni, rsilv}@inf.ufrgs.br

RESUMO

Hipertexto[1] desenvolvido sem metodologia de projeto adequada aumenta a probabilidade do usuário não assimilar as informações que está recuperando e ficar desorientado no hiperespaço, sem noção de quais nodos já visitou.

Esse artigo apresenta HSM (Hypertext Semantic Modeling) que faz uso do GSM (Generic Semantic Model) [4] para projeto de uma aplicação hipertexto. Modelos de dados semânticos tentam prover mecanismos mais poderosos para estruturação de objetos do que os providos pelos modelos tradicionais. O HSM captura aspectos estruturais (representação explícita de objetos, seus atributos e relacionamentos de classificação, generalização/especialização, agregação e agrupamento), facilitando a representação de dados complexos e o projeto navegacional de aplicações de hipertexto. A título de exemplo, é apresentado um caso de estudo, no domínio da fotografia, para ilustrar a aplicação dessa modelagem, o qual utiliza roteiros guiados indexados para implementar contextos de navegação.

[†] UFRGS e Departamento de Ciência da Computação da Universidade de Brasília

[1] Neste artigo não é feita distinção entre hipertexto e hipermidia.

1. INTRODUÇÃO

O grau de sucesso de uma aplicação hipertexto é relacionado com a habilidade do autor para capturar e organizar a estrutura de assunto em um domínio complexo de forma a torná-lo claro e acessível aos usuários [3]. Sem a utilização de um método formal, a produção de um hipertexto corre o risco de ser desenvolvida ao sabor do acaso ou da preferência pessoal do autor, principalmente no que tange à estruturação da informação e às ligações estabelecidas entre as diversas estruturas (nodos) [7]. Nesse caso a navegação entre páginas (nodos) pode tornar-se complexa, contribuindo para que o usuário sinta-se desorientado, sem saber que nodos visitou e com dificuldades para assimilar o conteúdo que o autor deseja transmitir.

O leitor navega através do hipertexto de acordo, normalmente, com seu interesse, grau de conhecimento e disponibilidade. A navegação em um hipertexto com muitas páginas é complexa e implica uma maior carga cognitiva para o leitor. Para minimizar esse esforço mental do leitor, durante o processo de manutenção da sua orientação, as aplicações hipertexto devem ser estruturadas de forma coerente e essa estrutura deve ficar clara. Uso de **caminhos**[1] predefinidos permite ao autor determinar uma ordem adequada de apresentação da informação, minimizando equívocos sobre o conteúdo do material por parte do leitor[11]. Se o conjunto de nodos é apresentado em uma ordem fixa, o caminho é denominado **caminho seqüencial** [11]. Roteiros guiados são caminhos seqüenciais criados para apresentar informação do hipertexto em uma seqüência predefinida, limitando a navegação aos nodos necessários para o entendimento de um certo assunto, em um certo nível de conhecimento[9].

[1] *Path* em Inglês; conjunto de nodos e ligações apresentados de acordo com um critério definido pelo autor e disparando ações predefinidas tais como tocar um som ou acionar um vídeo.

Segundo Schwabe e Barbosa, o mecanismo de **Roteiro guiado** permite ao autor criar um hipertexto complexo, mas compreensível para o leitor. Eles sugerem o conceito de **contexto de navegação** para "reduzir o aumento da carga cognitiva criado pelo grande número de opções de navegação tão característico de aplicações hipermídia, mas mantendo a flexibilidade" [9, p.5]. Contexto de navegação é um conjunto de nodos, ligações e informação contextual, que constitui um subgrafo de um hipertexto, cujos nodos recuperam um certo assunto de um certo ângulo e em um certo nível de conhecimento e garante a coerência do material apresentado, deixando o usuário confortável para explorar o material sem sentir-se desorientado ou perdido. Em geral inclui um caminho para sugerir o *melhor* próximo passo, mas o leitor deve ter possibilidade de outras escolhas. Pedaços de informação, apresentados em um contexto, adquirem significados mais claros e precisos, sendo mais fácil entendê-los.

Os contextos de navegação podem ser implementados enumerando-se o conjunto de nodos e ligações pertencentes ao contexto, via roteiro guiado ou via índice (que consiste em um nodo raiz e todos os nodos aos quais se pode ter acesso através dele) [9][1][5].

Ao combinarmos um roteiro guiado com um índice, obtemos um **roteiro guiado indexado**, no qual podemos ter acesso aos nodos anterior ao atual, o posterior, o inicial, o final do roteiro e qualquer outro nodo do roteiro, através do nodo raiz [9][1][5].

2. A MODELAGEM SEMÂNTICA PROPOSTA

Um **modelo de dados** é um conjunto de objetos lógicos usado para prover uma abstração de uma porção do mundo real [5]. A noção de modelo de dados é relativamente nova na área de hipertexto e é obviamente derivada da prática corrente na área de banco de dados [3]. As técnicas de modelagem de aplicações hipertexto, em geral, obtêm um modelo de dados através de uma

análise do domínio baseada em técnicas derivadas da área de banco de dados. Assim a RMD/RMM [1][5] e em certa forma HDM [3] utilizam a abordagem de entidade-relacionamento de Peter Chen[2] e HMT [7], OOHDM [10] e EORM [6] utilizam modelagem orientada a objetos.

Segundo Hull e King [4], a ênfase dos modelos semânticos iniciais era modelar acuradamente os relacionamentos que apareciam freqüentemente em aplicações típicas de banco de dados. Os modelos semânticos tentam prover abstrações mais poderosas do que as suportadas pelos modelos relacional, hierárquico e de redes, permitindo ao projetista pensar nos dados de modo correlacionado diretamente com a forma com que eles ocorrem no mundo real, facilitando a representação de conhecimento complexo e de suas inter-relações.

2.2 GSM - Generic Semantic Model [4]

Os componentes principais dos modelos semânticos são a representação explícita de objetos, seus atributos e dos seus relacionamentos; construtores de tipos para construção de tipos complexos; relacionamento *isa* e componentes de esquema derivado. O GSM permite a modelagem de aspectos estruturais e semânticos. Possui os seguintes tipos de dados: tipos abstratos (representado como um nodo triangular), subtipos (representados como um nodo circular) e tipos exibíveis[1] (representados como um nodo oval).

Tipos abstratos (objetos ou entidades) representam objetos físicos ou conceituais do mundo real, distintos dos seus atributos. Correspondem a abstrações de dados e não são exibíveis. **Subtipos** são subclasses (especializações) dos tipos abstratos. **Tipos exibíveis** são passíveis de serem exibidos em um vídeo ou impressora e possuem uma representação externa acessível ao usuário.

[1] *printable* em Inglês.

Podem ser utilizados para representar os atributos de um objeto. Os mecanismos de abstração disponíveis são: **classificação**, **generalização/especialização** (relacionamento *isa,* representado por uma seta em negrito, associa uma subclasse a uma superclasse), **agregação** (relacionamento *part-of*, representado por um x-nodo, indica que um objeto é composto por uma agregação de outros objetos da base de dados, formando um produto cartesiano) e **agrupamento** (relacionamento *member-of*, representado por um ✳-nodo, representa um conjunto finito de objetos de um mesmo tipo). Os atributos de classe ou objetos podem ser **monovalorado**, **multivalorados** ou **derivados** a partir de uma expressão ou regra e com base nas informações armazenadas na base de dados. (Veja Figura 1).

Figura 1 - Primitivas do Domínio

Os subtipos do GSM podem ser **definidos pelo usuário** ou **derivados** a partir dos dados da base de dados, através da especificação de operações na base de dados. [4] apresenta um exemplo de um esquema conceitual de banco de dados com atributos e subtipos derivados.

As modelagens semântica e orientada a objetos apresentam algumas semelhanças. A modelagem semântica permite a expressão dos aspectos semânticos dos dados, sendo forte no encapsulamento de aspectos estruturais. Tipicamente incorporam mais construtores de classe do que a modelagem orientada a objetos e o conceito de herança de atributos é limitado ao relacionamento superclasse-subclasse.

Já a modelagem orientada a objetos permite expressar as relações estruturais e dinâmicas (funcionais ou comportamentais), sendo forte no encapsulamento de aspectos dinâmicos dos objetos. O conceito de herança de métodos pode ocorrer entre diferentes objetos e os atributos de objetos ou classes podem ser apenas monovalorados ou multivalorados.

A modelagem semântica pode ser vista como uma extensão do modelo relacional mas, para capturar o máximo possível da semântica de um esquema semântico original, devem ser incluídos no esquema relacional dependência de chave e dependência de inclusão[1][4][8]. **Dependência de chave** estabelece que o valor de um (ou mais campos) de uma tupla determina os valores dos demais campos da tupla. **Dependência de inclusão** determina que todos os valores que ocorrem em uma ou mais colunas de uma relação também ocorrem em alguma(s) coluna(s) de outra relação.

2.3 A Abordagem HSM

HSM é composta de cinco etapas: análise do domínio, projeto navegacional, projeto de conversão de protocolo, projeto de interfaces com o usuário e implementação. A **análise do domínio** consiste na representação dos objetos, classes, subclasses, superclasses e associações com base no GSM, obtendo-se como produto o modelo de objetos. O **projeto navegacional** consiste na agregação de novos relacionamentos e objetos que estabelecem caminhos desejados, do ponto de vista de navegação e que não foram capturados no modelo de objetos. É feita a explicitação das mídias que farão parte da aplicação final e um mapeamento dos objetos conceituais em objetos navegacionais. A Figura 2 contém as primitivas de acesso. A seguir são definidos os contextos de navegação e os pontos de entrada da aplicação. Os contextos de navegação são do tipo arbitrário,

[1] *key dependency* e *inclusion dependency* em Inglês.

computados ou de sessão[9]. O **projeto de conversão de protocolo** consiste no mapeamento dos objetos navegacionais em objetos no sistema de hipertexto a utilizar na implementação da aplicação. O **projeto de interfaces com o usuário** consiste no projeto dos *layouts* de tela, definição da aparência dos objetos e especificação das identificações visuais, em função da plataforma de hardware e software a ser utilizada.

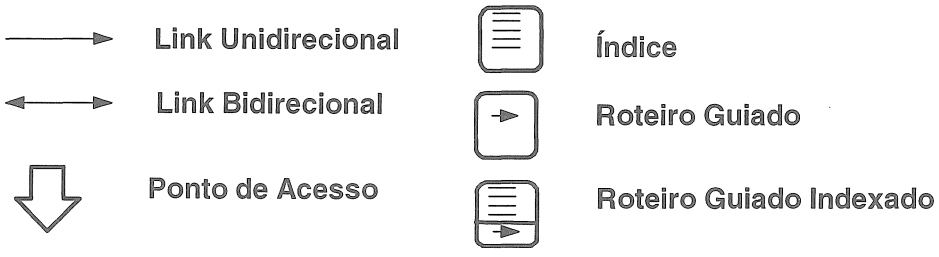


Figura 2 - Primitivas de Acesso

3. EXEMPLO DE USO

Foi elaborado um hipertexto com conceitos sobre equipamentos fotográficos, filmes e técnicas de fotografias, destinado ao fotógrafo amador e implementado em HTML, em rede SUN, disponível via URL no http://tucano.inf.ufrgs.br/~mladeira/fotografia/home.html.

A Figura 3 apresenta o modelo de objetos obtido. Foram modeladas as classes fotógrafo, máquina fotográfica, foto, componentes, acessórios e filme.

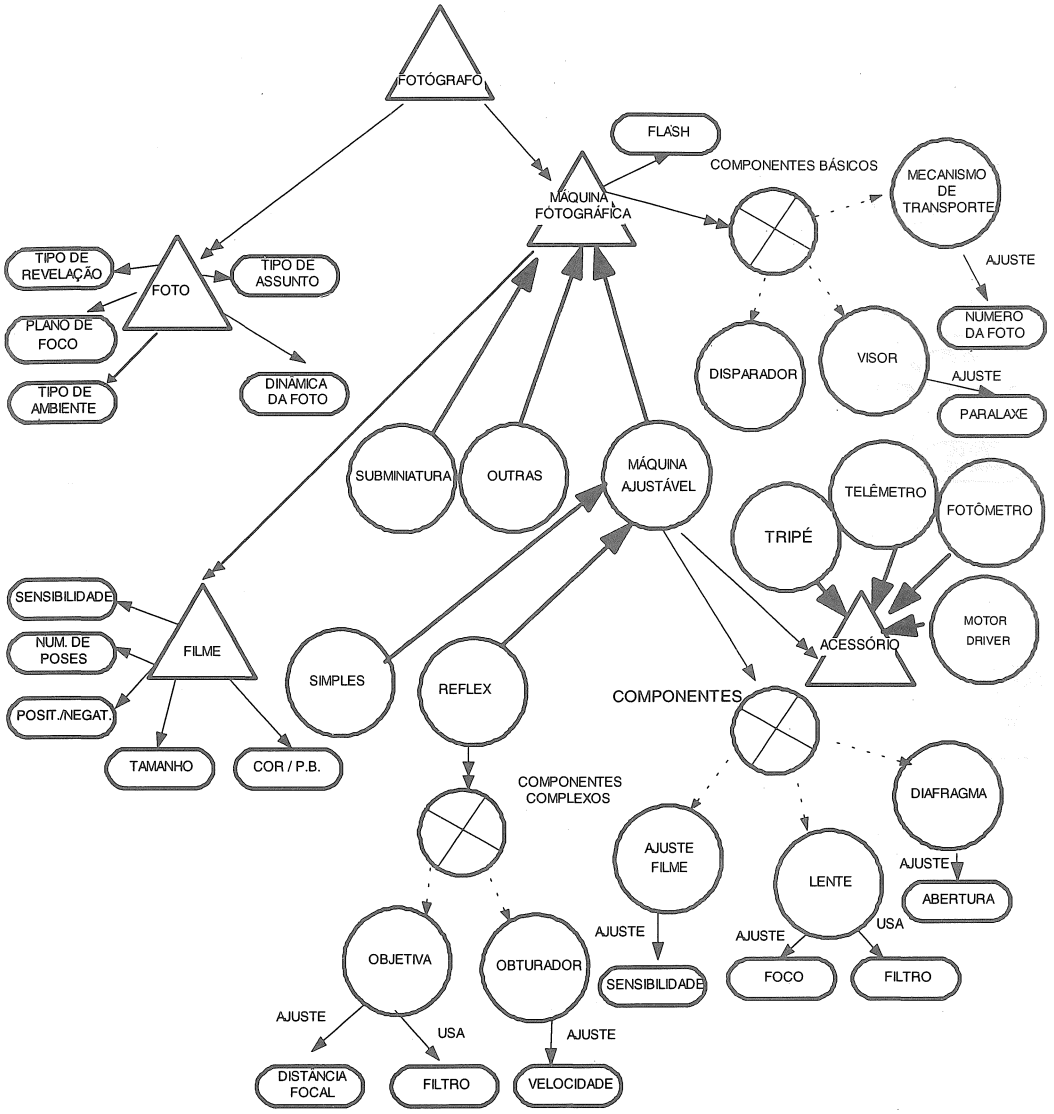


Figura 3 - Modelo de Objetos para Fotografia

A Figura 4 apresenta o modelo de navegação, com a inclusão de relacionamentos (linhas tracejadas) tais como máquina fotográfica com máquina fotográfica, foto com componentes e especificação da mídia a ser utilizada para representação, em geral texto e imagem.

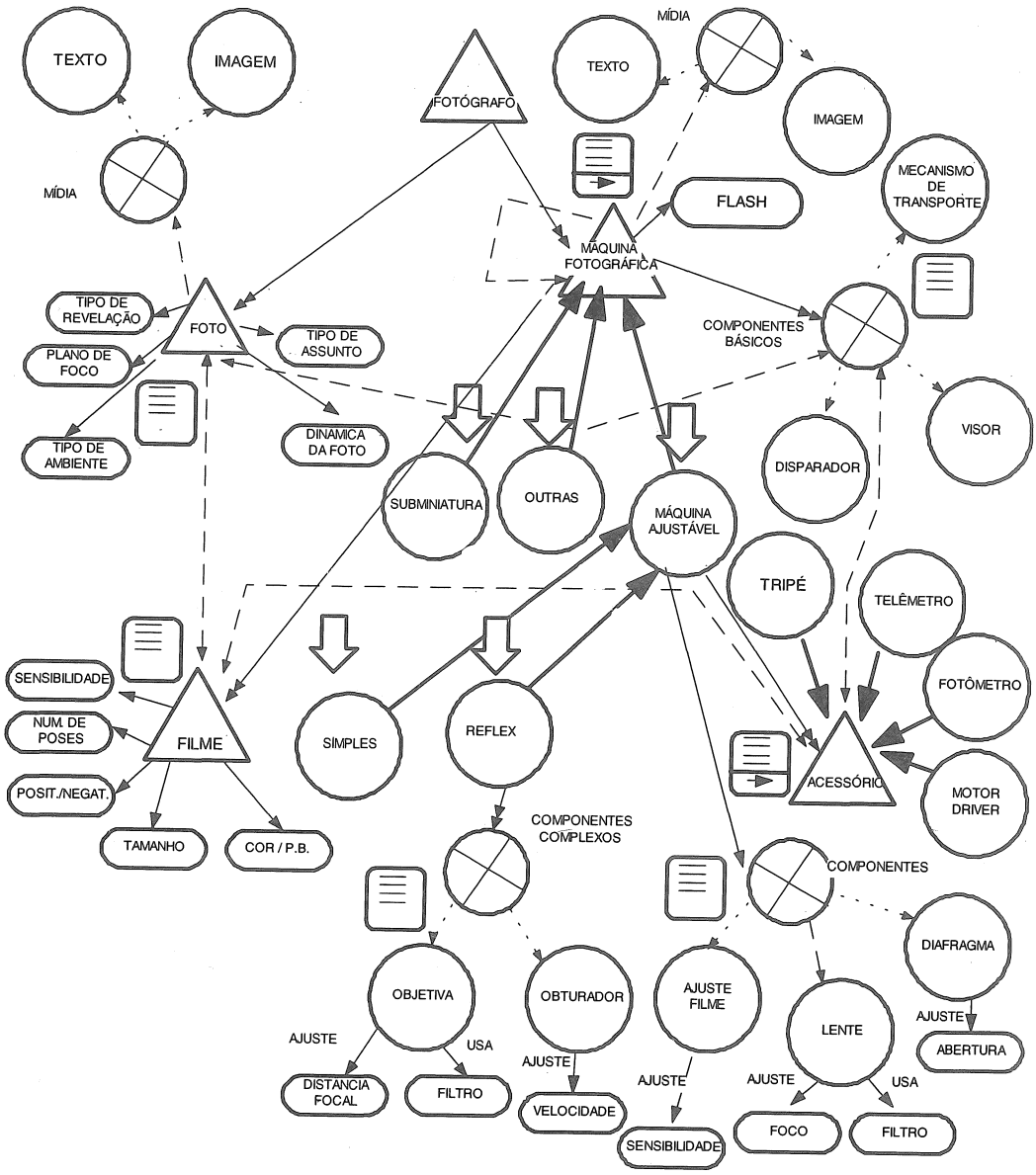


Figura 4 - Modelo de Navegação para Fotografia

Foram considerados quatro contextos, função do tipo de máquina: **subminiatura**, **ajustável simples**, **ajustável reflex** e **outras**. Para os três primeiros contextos foram criados roteiros

guiados indexados compostos dos seguintes itens: a) *características da câmara*, b) *cuidados ao fotografar*, c) *tipo de foto*, d) *componentes*, e) *acessórios* e f) *filme*. Para o contexto *outras* foi criado um roteiro guiado indexado composto de *máquina de estúdio*, *polaroid* e *digital*.

Para facilitar o mapeamento entre objetos conceituais do domínio e objetos navegacionais pode-se utilizar as diretrizes a seguir. A objeto do tipo **agrupamento** corresponde um índice de classe; aos atributos de um objeto ou a um objeto do tipo **agregação** corresponde um roteiro guiado; e aos membros de uma classe corresponde um roteiro guiado indexado ou um índice de classe. Um **índice de classe** é um tipo especial de índice, no qual todos os nodos aos quais se pode ter acesso correspondem a instâncias da classe [9]. Um relacionamento monovalorado pode ser implementado via ligações bidirecionais no hipertexto. Já um relacionamento multivalorado pode ser implementado via um roteiro guiado, índice ou mesmo um roteiro guiado indexado. Contextos de navegação computados podem ser derivados das ligações do modelo de objetos e roteiros guiados podem ser contextualizados por tipo de objetos. No exemplo de aplicação os roteiros guiados foram contextualizados pelo tipo de câmara.

O projeto de conversão de protocolo considerou a implantação da aplicação *fotografia* em HTML, com browser Netscape 2.0 b6, em uma rede SUN. Devido ao uso de contextos navegacionais foram implementados menus contextualizados. Assim a informação a eles associada ficou distribuída pelos contextos. Para evitar a construção de páginas muito pequenas, as informações associadas aos atributos de um objeto, em alguns casos, não foram implementados através de roteiros guiados.

No projeto de interface com o usuário, a fim de reduzir a carga cognitiva do leitor, foram implementados menus (de mesmo *layout*) ao final de cada página dos roteiros guiados indexados,

com a opção *outros contextos* (representada por ícones de cada tipo de câmara) e as páginas *primeira, anterior, posterior* e *última* do contexto atual. Esses menus são de contexto, ou seja, por exemplo, se o usuário está no item *características da câmara* do contexto *subminiatura* tem acesso ao item *características da câmara* dos outros contextos, através da opção *outros contextos*. Para facilitar a identificação do contexto, cada roteiro guiado indexado possui uma cor de fundo diferente, visível a partir do browser Netscape 1.1+ e a indicação do nome do contexto através da marca *title*. Como facilidade adicional disponível para o leitor, existe um mapa "*clicável*" derivado diretamente do diagrama GSM.

4. CONCLUSÕES

HSM difere da RMM [1][5] e do HDM [3] principalmente devido a ênfase dada ao projeto navegacional e ao projeto de interfaces com o usuário. Além disto, RMM e HDM utilizam análise do domínio baseada em entidade-relacionamento enquanto no HSM é baseada em modelagem semântica. HSM difere de HMT[7], OOHDM[10] e de EORM[6] principalmente porque eles utilizam análise do domínio baseada em modelagem orientada a objetos. HD1 [8] também utiliza modelagem semântica, mas a aplicação hipertexto é desenvolvida no topo de um banco de dados semântico experimental, denominado SABEREL, em desenvolvimento pela IBM. HSM não se apoia em nenhum banco de dados.

O uso do HSM mostrou ser viável para o projeto de hipertextos complexos, facilitando o projeto navegacional e a própria estruturação do hipertexto. Obviamente a estruturação do domínio em objetos e a explicitação de seus relacionamentos facilitam a especificação dos nodos e das ligações do hipertexto. O autor de uma aplicação hipertexto que utilize um modelo de dados baseado em modelagem semântica tem mais facilidades para a modelagem das relações

estruturais de objetos complexos, além de dispor de um conjunto mais rico de construtores de classes do que os providos pela modelagem orientada a objetos.

O autor também pode utilizar uma versão simplificada do diagrama GSM para elaborar um mapa "*clicável*" que apresente um visão global da estrutura navegacional, auxiliando a orientação do leitor. O próprio projeto navegacional é facilitado visto que o autor pode fazer um mapeamento entre objetos conceituais do modelo de dados e estruturas navegacionais conforme ilustrado na seção 3.

5. REFERÊNCIAS BIBLIOGRÁFICAS

1. BALASUBRAMANIAN, P; ISAKOWITZ, T. & STOHR, E. Designing Hypermedia Applications. In **Proceedings of the 27th International Conference on System Sciences**, Maui, Hawaii. New York: IEEE Press, 1994, p.354-365.

2. CHEN, P. The Entity-Relationship Approach - Toward a Unified View of Data. **ACM Transactions on Database Systems**, v.1, n.1, 1976, p.9-36.

3. GARZOTTO, F.; PAOLINI, P. & SCHWABE, D. HDM - A Model-Based Approach to Hypertext Application Design. **ACM Transactions on Information Systems** v.11, n.1, Jan. 1993, p.1-26.

4. HULL, R. & KING, R. Semantic Database Modeling: Survey Applications and Research Issues. **ACM Computing Surveys** v. 19, n.3, Sept. 1987, p.201-260.

5. ISAKOWITZ, T.; STOHR, E. A. & BALASUBRAMANIAN, P. RMM: A Methodology for Structured Hypermedia Applications. **Communications of the ACM**, v.38, n.8, Aug. 1995, p.34-44.

6. LANGE, D. An Object Oriented Design Method pof Hypermedia Information Systems. **International Conference on System Sciences**, 27, Jan. 4-7 1994. Maui, Hawaii. (Proceedings ...). New York: IEEE Press, 1994.

7. NEMETZ, F.; LIMA, J. V. DE & BORGES, A. C. Uma Técnica para Modelagem de Aplicações Hipermídia e um Exemplo de Uso. In Anais do **XXII SEMISH / XXI Conferência Latina-Americana de Informática**. SBC/CLEI, 1995, p.261-274.

8. SCHNASE, J. L.; LEGGETT, J. J.; HICKS, D. L. & SZABO, R. L. Semantic Data Modeling of Hypermedia Associations. **ACM Transactions on Information Systems**, v.11, n.1, Jan. 1993, p.27-50.

9. SCHWABE, D. & BARBOSA, S. **Navigation Modeling in Hypermedia Applications**. Rio de Janeiro: PUC-Rio - Departamento de Informática, 1994, 20p. (Tech. Report).

10. SCHWABE, D. & ROSSI, G. The Object-Oriented Hypermedia Design Model. **Commun. of the ACM**, v.38, n.8, Aug. 1995, p.45-46.

11. ZELLWEGER, P. T. Scripted Documents: A Hypermedia Path Mechanism. In **HYPERTEXT'89 Proceedings**. Pittsburgh: ACM Press, Nov. 1989, p.1-14.

ACAM - an Active Context-based Authoring Model

Judith Kelner,
D. F. Hadj Sadok, Ricardo Martins.
Departamento de Informatica,
Universidade Federal de Pernambuco,
Cidade Universitaria - Recife - Pernambuco,
Caixa Postal - 7851,
CEP - 50732-970,
Brasil.

E-mail: {jk, jamel, rmas}@di.ufpe.br,
Tel: +55 (81) 271-8430,
Fax: +55 (81) 271-8038

Abstract

This work examines existing authoring models and methodologies. A new interactive, distributed, active, context-based authoring model is proposed.

The elaborated hypermedia model for distributed authoring is based on the concept of *contexts* and combines different authoring paradigms. Contexts define presentation policies and perspectives and offer a high level structural organisation of both the information and the authoring process.

Another contribution to research in the area made by this model, is its use of intelligent cooperating active agents to support run-time reference resolution, dynamic loading of applications and user interaction. The model adopts a modular multi-layered distributed architecture supported by the Sun's new Java object oriented language. The model makes clear division of the authoring process into two main stages, namely an authoring in the large and an authoring in the small phases.

Introduction

This work looks at the process of creating multimedia presentations, in other words, *multimedia authoring*. Continuing research is undertaken to develop new multimedia interactive models and many professional authoring products have already reached the market. Examples of these include Macromind's Director [Mac90], the Authorware package described in [NO94] and Harmony [ea91]. One major drawback of existing models and systems is their lack of high level adaptive use of the environment and distribution.

Hypermedia authoring involves the extra step of linking the authored information using a graph structure where links can be navigational, structural, semantical, etc. There is a wide spectrum of applications for authoring including: Interactive Multimedia Projects: describing systems, products, etc, Computer Based Training (CBT), Kiosks, Animations, Interactive TV and the Film industries, etc. Other applications include Training and Education also known as Teletraining, Simulation and Games, Sales and Advertising, Office Information Systems, Enginnering Documentation, Electronic Publishing, Medical Applications. More information on these applications may be found in [MG95].

Review

A more comprehensive review of existing Hypermedia and Multimedia models can be found in the authors' technical report[dASeKLS95]. This survey shows that existing hypermedia models are mostly generic and offer limited support to authoring and even those that do, such as HMCARD[KAS94], often lack the necessary distribution and collaboration support and are usually far from being intuitive to learn how to use.

The discussion in [Pei95] includes HDM which is of particular interest to this work. This is an authoring model [GPS93] that allows the generic planning of hyper-documents. Its use of the concept of link perspectives, later described, is retained into the Active Context-based Authoring Model (ACAM).

Other *platforms*, such as those described in[ea93b] and [ea93a], support the use of re-usable and configurable active objects for the authoring of applications. While the authoring environment IMP (Interactive Multimedia Platform) suggested in [BB95] makes an attempt to address the entire spectrum of multimedia presentation it makes little or no effort for an analysis of the modelling of authoring systems.

Hypermedia based authoring systems include Harmony [ea91], Athens [VM93] and CMIF[1] [Bul91]. They support event driven functions used for building applications . These models together with other PC based ones such as Apple's Quicktime [Wan93] are mono-user systems and offer no

[1]The CMIF authoring model from CWI is based on the Harmony model.

	Dexter	Proxhy	Acqua	Hypercard	HDM	HM	OOHDM	Authorware
Concepts Simplicity	•	•	•	•	•	o		•
Semantics			o	o	o	o	•	
Application Modularity			o	o	o	•	•	o
Hypermedia Modeling	o	o	o	•	•	o	•	
Application Modeling			o	o	•	•	•	o
Cooperative Modeling			o					
BottomUp/TopDown					o	•	•	•
Security			o	o				
Version			•					
Design Reusability						•	•	

Table 1: Comparison of some Hypermedia Models: **none** not found • found o insufficient

distribution support. Other standardisation efforts from MHEG (Multimedia and Hypermedia Expert Group) [MG95] and HyTime[ISO92][2] do not take into consideration the distribution aspects. This may change through the work of IMA [IMA93] nonetheless.

A brief comparison of some authoring models and systems is shown in the Table 1.

Requirements for Authoring Models

A more detailed list of hypermedia and multimedia application requirements can be found in the MHEG document [MG95]:

- system transparency to users, composition of multimedia data in time and space, synchronisation (elementary, chained, cyclic and conditional) and automatic adjustment;
- intuitiveness, simplicity of use and a short learning curve, object orientation and partial re-use of structures, information and projects;
- an authoring environment that makes it quick and easy to produce different levels of applications: simple presentations, hypermedia programs, and more sophisticated interactions with *answer judging*, feedback, and record keeping;
- *universal* access to retrieval services such as gopher, WWW, WAIS and multimedia database retrieval, support for international interchange standards such as MHEG and HyTime, real time interchange and presentation;
- a presentation service: with functions for sequential presentation, branching, conditional, query support, etc;

[2] HyTime is the official nickname given to the ISO/IEC international standard 100744: 1992, also known as the Hypermedia/Time-based Structuring Language"

- exploration, communication, media annotation and automatic generation of certain types of links, aggregation - an abstract mechanism used for referring to a collection of objects using a unique identifier;
- configuration, navigation and different interaction modes, quality of service (QoS) management and ability to build management policies;

Presentation and User Interaction Requirements

A number of the existing multimedia platforms, have little dealt with user interaction aspects. This is mainly due to the fact that only few of these are oriented towards authoring. Events are defined which represent asynchronous run-time operations as event functions. These include user interaction, system signals, etc. One of the novelties of the MHEG standard over existing multimedia interchange formats is its identification and encoding of user events as objects just like data objects including text, graphics, etc.

Context

The problem with using hypertext style of information access and retrieval is a *cognitive overloading* which could even lead the user to a loss of orientation. Although the term "context" is very generic, it is possible to identify in hypertext systems, some types of navigational contexts [SB95] as: arbitrary, topics derived, link derived, etc. These are explained later in this document. Contexts can be viewed from two different angles:

- an authoring context: enables the author to organise information into topics, themes or subjects, etc. This way, contexts are used to enforce intuitive authoring information classification activities.
- a user context: can be used to define presentation policies within a company, a group, etc.

Modelling Authoring Systems

A model offers an authoring language and makes development a structured task independent of the application. A major drawback identified in existing authoring systems is that they deal with the authoring of multimedia information and ignore the conceptual phase of a *project* design. This research considers both stages:

- *project authoring or authoring in the large:* where organisational and conceptual aspects of a project are considered. The project is viewed from a very different point of view. The semantics of the links among the elements of the project are referred to as *"application links"* since they differ from one application to the next;

- *authoring in the small:* illustrates the refinement and instantiation of the components of the project previously identified. This stage implements traditional authoring mechanisms such as synchronisation, media editing and linking, and presentation.

In order to present some of the aspects of the ACAM model, an example is used throughout this document. It consists of designing an authoring application describing the State of Pernambuco[3]. The application could start with a project authoring view that shows the main characteristics and subjects characterising this State. The presentation of which could be divided into several areas dealing with the Political, Economical, Historical, Cultural, Geographical and Touristical characteristics of this region of Brazil as shown in figure4. At the project level, a graph with *application links* can be used to show the way these components are related to each other. This mechanism was first used in the HDM model [GPS93]. An example of such a relation would be the link "events" that spans between the touristic and cultural information.

Authoring Paradigms

Authoring should be a systematic and a well structured process. There are very few techniques specifically designed for the authoring of hypermedia documents. A great deal of these are empirical techniques [GPS93]. Currently there is no completely "point-and-click" authoring system; some knowledge of algorithm design and heuristic thinking is required [Sig95]. As it is, authoring can be seen as an enhanced mode of programming where although the author neither has to learn a programming language nor has to use a specific Application Programming Interface (API), a basic understanding of how programs work is needed[4].

1. time and flow based iconic representation such as Authorware;
2. card based such as Apple's HyperCard;
3. frame paradigm;
4. script based systems such Hypertalk used in Apple's HyperCard and Toolbook from Asymetrix;
5. structure editing systems as described in the paper[LHB93];
6. frame direct timing relations paradigm used in Harmony[ea91] and Videobook[ROK90];
7. the cast score scripting paradigm;
8. collection based as in HM-Card [KAS94];

The ACAM user interface adopts adopts a combination of the flow icon based, collection and script paradigms with support for structure editing and run-time execution objects.

[3] Pernambuco is one of North Eastern states of Brazil

[4] A detailed description of existing paradigms is found in[dASeKLS95].

The ACAM Model

ACAM defines the four main modules shown in figure , namely, presentation, authoring, hypermedia and distribution.

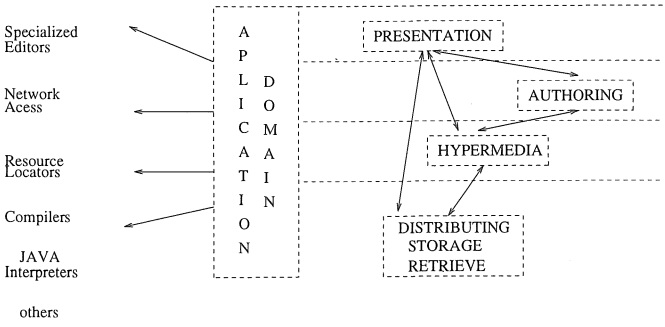


Figure 1: ACAM's Modules

Presentation

This layer executes the author's model designed during the authoring process, through the use of ACAM's mechanisms, services and the interaction with other tools and systems from its environment that are external to ACAM. Its main tasks include the execution of *pre-defined* synchronisation aspects, layout and content control, user interaction and event management. In order to cater for applications such as computer based training, (CBT) where it is often necessary for interaction between users, this module supports *distributed presentation scenarios.*

Authoring

Models the application, starting with a conceptual abstract model and refining this through the instantiation of its components. ACAM divides the authoring process into two stages: authoring in the large and authoring in the small.

Hypermedia

It is the hypermedia module that is responsible for the hypertextual aspects of the application. The following link types have been defined: application links, links of perspective, synchronisation links, conditional (flow control) links etc. This module is also responsible for the control and manipulation of the semantics associated with authoring contexts, version control, access control and security. Other services of this layer include the support of cooperative work and most of

all the identification (object referencing) and transparent location of multimedia information objects.

With the emergence of new multimedia representation standards such as the *Multimedia and Hypermedia Expert Group* MHEG[MG95], already dealing with important multimedia issues including synchronisation, event manipulation, object identification, etc, there is no longer a need for a separate object layer. The hypermedia layer also handles links among objects that have semantics associated to their role in the model. Examples of these are synchronisation objects (e.g audio synchronisation object, etc), presentation objects, display objects, and monitoring objects.

Distribution, Storage and Retrieval

The new design architecture offered by the use of active *smart* [5] agents are being used instead of current rigid protocol based applications that lack flexibility and real distribution. Security concerns remain however, a major obstacle to overcome. A number of other platforms also support the use of re-usable and configurable active objects for the authoring of applications. [ea93b] and [ea93a] adopt the use of active objects. The MIT [ea94] model used in VuSystem, goes further in that it takes into account the distributed environment. VuSystm uses an extended TCL but has no specific support for user interaction. Examples of languages used by smart agents include Telescript, WAVE [Sap93], JAVA[Sun95b] etc.

Functional Architecture

Although an organisation of software modules into layers as the ones shown in figure , usually implies that one level makes use of the services of the layer below it, the layering shown here depicts a purely abstract hierarchy where higher levels are more abstract than lower ones.

The openness of the proposed model is shown with ACAM's integration to its environment. This integration is made with all the services offered externally. Such services would normally include, dedicated editors and presentation tools, communication software and tools such as WWW, FTP, WAIS, Java interpreters, MHEG engines, etc.

An Authoring Methodology

The design activity can be described as a two dimensional space between the author's mental model and the supported authoring methodology. Authoring oscillates between the two axis as shown in figure.

Figure 3 shows the four stages of the adopted authoring methodology:

[5] The term smart is used instead of intelligent to describe these agents in order to differentiate it from artificial intelligence based systems.

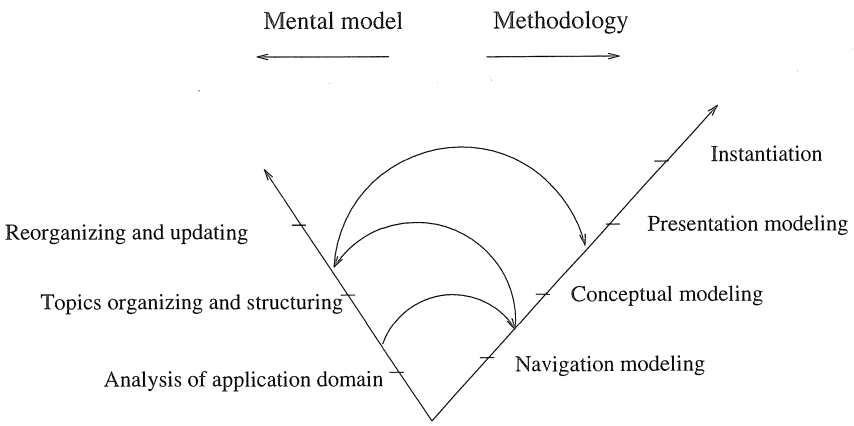


Figure 2: Authoring Design Activity

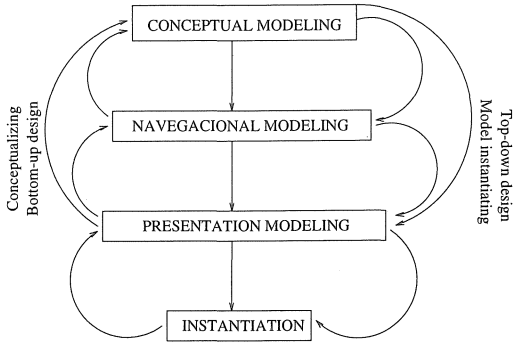


Figure 3: Hypermedia Applications Development Methodology

- conceptual modeling - an open authoring stage that allows the modelling of real world situations. It defines relations among topics, subjects or themes;
- navigational modeling - oriented towards the manipulation of the conceptual model previously created; it defines the navigational aspects through the promotion and election of parts of the topics, subjects or themes to the status of nodes; and the promotion or election of some of the relations to the status of links referred to as "application links";
- presentation modeling - at this stage is defined the presentation interface and user interaction aspects. Different aspects are considered, including static aspects such as application layout, dynamic aspects with synchronisation, and the behaviour of authoring objects under the effect of external user interface events;

- instantiation - multimedia data is created and associated to the elements of the presentation according to the pre-established author's model.

A consideration here, would be for the system to automatically extract or deduce the associated navigational and conceptual models. This part of the work is left for further research.

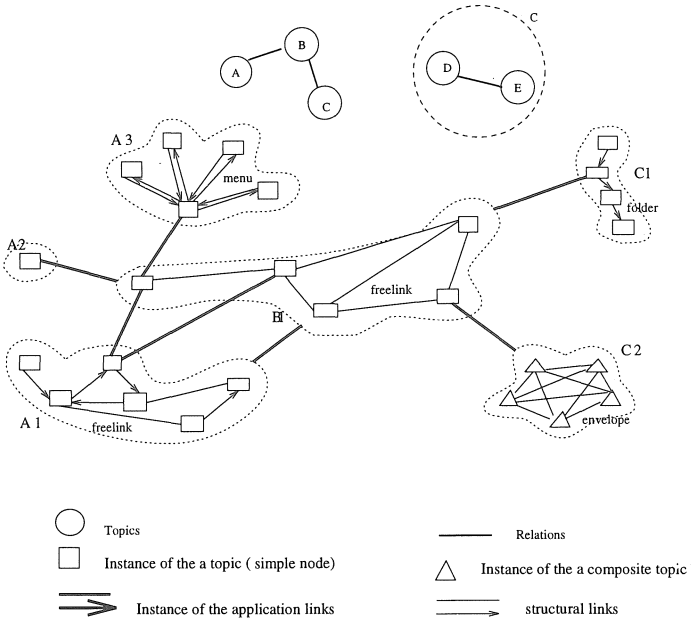


Figure 4: Hypothetical Conceptual Model

Figure 4 shows the conceptual model with the hypothetical topics: A, B and C. Where C, is considered a complex theme, which has sub-themes: D and E. Therefore topic C consists of an internal structure as well as content.

The topics (subjects or themes) shown in the figure, are considered as nodes in the second stage of authoring. They are represented by arcs in the conceptual model graph; these relations, which in the second stage could be application links showing relations among related topics and corresponding to different subjects, are represented using thicker lines. The content instances for a given topic are hyperdocuments rather than simple nodes. The reason behind this choice, is the ability to provide the author with complete freedom when instantiating the hyperdocument which can be a simple object, constructed by other nodes or complex with an associated navigational structure. Figure 4 shows that for each of the subjects treated there are few associated instances - A1, A2 and A3. These are instances of A; similarly B1 is an instance of B and C1 and C2 are instances of C. Each subject instance could be linked to other instances from other topics according to the pre-established abstract model. This way an applicative link between instances

A and C cannot be allowed since it is not specified in the model.

Structure of a Topic Node

An ACAM topic node consists of:

- *content:* information that characterises a topic, or more precisely, determines the behaviour of its instances such as: perspective links, security and access control constraints, structural navigational styles and other attributes.
- *internal structure:* used in order to represent complex themes subject to ZOOM-IN and ZOOM-OUT operations shown in figure5.

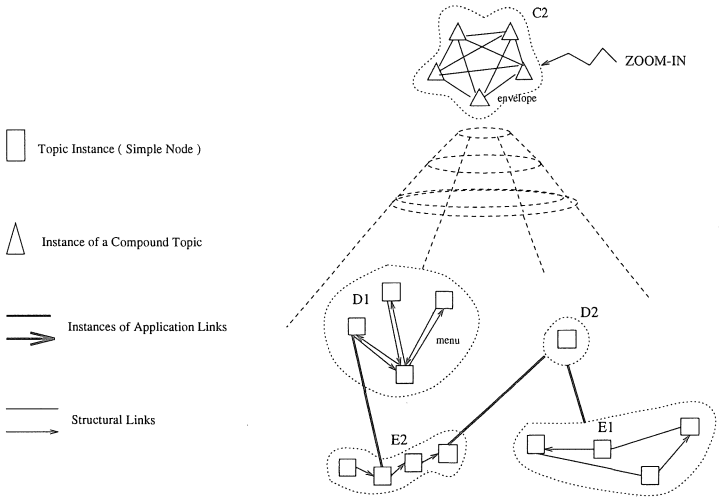


Figure 5: ACAM's use of Collections and Zoom Operations

Phase 1 - Authoring in the Large

Allows the creation of a model for a given specific application domain, see figure3. It defines the topics or themes, their relationships, attributes and operations; which are considered important during requirements analysis. In this initial stage, the domain application has been modelled as an abstract conceptual graph where vertexes and edges represent topics and their relationships respectively. In figure 4, the relationships among Tourism and Culture follows intuitively. This relationship permits a possible binding among related topics instances during the authoring in the small. Looking at the information system of the Pernambuco State, one sees that the themes are very generic. This way, one needs to create sub-themes that show more specific topics, but still belong to one or more of the abstract themes. Figure 6 shows the way ACAM handles

topics and themes. Figure6 shows a conformance among the classification of the topics "Tourism and Culture" and the model. In other words, the author is required to follow the hierarchical structure shown in figure 6, and may further extend it with relationships among sub-topics.

Often the more abstract themes hold generic attributes, while simple ones have more specific characteristical information associated to them. The definition mode of this attributes and perspectives consist of:

- declaring the perspectives of the topics, e.g., in Geography we have text, graphics and audio;
- defining attributes identifying topics, e.g. the theme Tourism can be seen to contain the attributes: title, city-name, general-information and post-card;
- defining perspectives of the attributes which are sub-sets of the perspectives of the topics;
- not declaring anything, that is, make the model very abstract and as generic as possible;

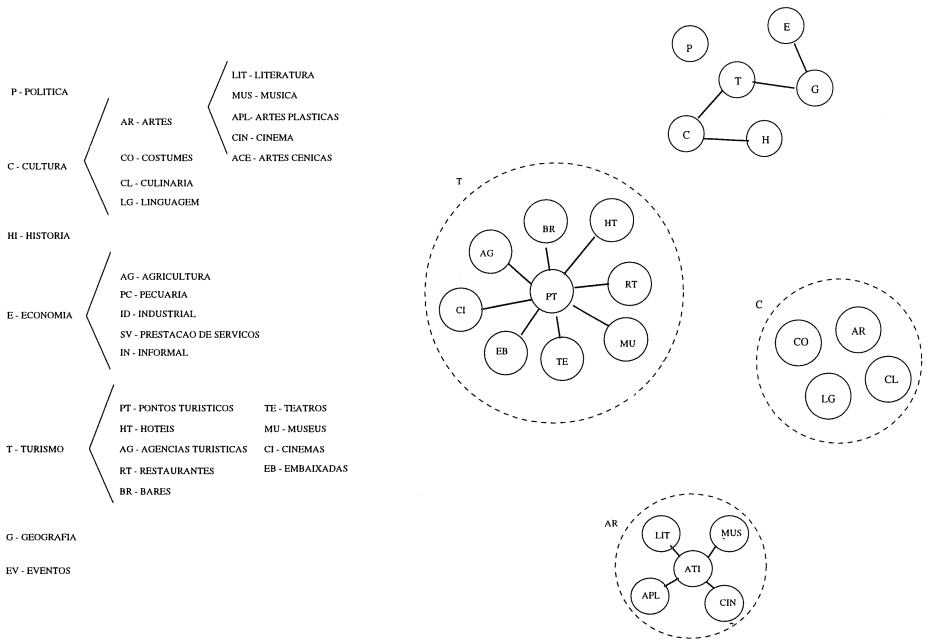


Figure 6: Practical Conceptual Model

Phase 2 - Navigational Modeling

Following the conceptual schema resulting from the previous phase, the author begins b ilding a vision over the hypermedia system, creating the navigational basic objects based on the topics

and relations defined in the conceptual modelling phase. Such objects include: nodes, links (arbitrary links, topics derived links, relation derived links and links derived from the internal structure).

Both conceptual and navigational modelling are going to guide the building of a specific multimedia authoring application. This new stage is known as authoring in the small within ACAM. It defines the final stages of the authoring process responsible for modelling the presentation and instantiating application information components.

Phase 3 - Presentation Modelling

The presentation layer deals with issues such as the application layout and the synchronisation of objects built during the instantiation stage. This third stage of the authoring process models the application presentation aspects in time and space. The following describes the authoring aspects of presentation modelling:

- *static aspects:* that deal with the application object layout information. The model adopts the use of user definable templates as background guidelines for all or part of the application objects and topics.
- *dynamic aspects:* represent presentation functions that deal with the synchronisation of multimedia objects built during the instantiation phase.

Other dynamic presentation aspects include support for the dynamic adaptation to resource availability that can be provided by invoking methods on associated objects at run time. This mechanism allows the building of different presentation policies and schemes. CMIF[Bul91] is an example of a system that adopts this mechanism where each component can be represented by a number of media (such as video versus audio and text, etc). The presentation layer selects which option to use depending on resource availability during run-time.

The authoring model ACAM defines a number of related editors that control how the authoring paradigms can be manipulated to suit the author's specific application:

- *time based flow editor:* used to offer a hierarchical synchronisation service;
- *a temporal editor:* responsible for the specification of sophisticated inter-media synchronisation;
- *script editor:* programs the events declared in the flow editor.
- *layout editor:* is responsible for the definition and manipulation of object layout and formatting aspects;
- *content editors:* represent all media dependent dedicated editors such text, bitmap, graphics, and other editors.

Phase 4 - Instantiation

It is responsible for the creation of multimedia data and its association to the elements of the presentation according to the pre-established author's model and templates. ACAM defines different types of links including navigational, of perspective, application links, etc. These have the structure defined by the attributes: identifier, type, name, entity of origin, entity of end, symmetry, method, etc. Access structures are used to define the way a user may access the authored information. These include: routes, indexes, user queries and associative access.

Conclusion and Future Work

This paper presented a hypermedia authoring model based on the concept of context based authoring and which uses underlying communicating agents to provide adaptive run-time reference resolution, access and loading of information and software applications, and distributed processing. Although distributed platforms have been among us for a long time now, they have received limited commercial support. It is the authors view that the future lies in the use of smart agents and languages for building distributed applications.

With the distribution of the whole model a number of advantages have been achieved including load balancing, support for users with limited resources such as mobile users, redundancy of the components which increases system availability, etc.The adoption of the Java language brings a number of advantages as far as Internet access is concerned. The author may dynamically load referenced external objects using this capability supported by the Java language during run-time. Last but not least, an *interactive* demonstration of the prototype will be made available through the Internet to remote users. This may be accessed through Internet browsers that support a Java interpreter such as the netscape WWW browser and the HotJava[Sun95a] browser from Sun MicroSystems.

One of the positive aspects of ACAM is that is has been designed to be generic enough to capture a number of applications and systems, to include new functionality and modules as these are identified and its openness through the run-time interpretation of Java code. There are, however a number of remaining open issues that one would need to see them dealt with. An important one is the management and use of Quality of Service (QoS) controls. Although, the model supports a high level adaptive use of the environment through the definition of the concept of perspectives, there is no explicit use made of QoS parameters.

Support for cooperative authoring is another aspect of authoring that is lacking from the current model. This, clearly is an important characteristic and needs further consideration in the coming stages of this project.

Acknowledgements

This work has been undertaken as part of the Project SAM (Sistemas de Autoria Multimedia), sponsorship by the CNPq/Brasil of the ProTem-II program.

References

[All95] Michael W. Allen. *Authorware Models for Instructional Design.* Prentice-Hall, 1995.

[BB95] John Bates and Jean Beacon. Supporting interactive presentation for distributed multimedia applications. *Multimedia Tools and Applications - An International Journal*, 1(1):47–78, March 1995.

[Bul91] D. Bulterman. A structure for transportable dynamic multimedia documents. In *USENIX Multimedia for Now and the Future*, 1991.

[dASeKLS95] Ricardo Martins de A. Silva e Kleber Leal Sales. Um estudo sobre modelos e sistemas de autoria em multim/'idia e hipermídia. Technical report, Departamento de informática da Universidade Federal de Pernambuco, 1995.

[ea91] K. Fujikawa et al. Multimedia presentation system harmon with temporal and active media. In *USENIX: Multimedia for Now and the Future*, pages 1–10, 1991.

[ea93a] S. Gibbs et al. Application construction and component design in an bject oriented multimedia framework. *Visual Objects - University of Geneva*, 1993.

[ea93b] S. Gibbs et al. Data modelling of time-based media. *Visual Objects - University of Geneva*, 1993.

[ea94] D. Tennenhouse et al. Software oriented approach to the design of media processing environments. In *Proceedings of the first IEEE International Conference on Multimedia Computing and Systems*, 1994.

[GPS91] Franca Garzotto, Paolo Paolini, and Daniel Schwabe. HDM - A Model for the Design of Hypertext Applications. *Proceedings of Third ACM Conference on Hypertext*, pages 313–328, December 1991.

[GPS93] Franca Garzotto, Paolo Paolini, and Daniel Schwabe. HDM - A model approach to hypertext application design. *ACM Transactions on Information Systems*, 1(11):1–26, January 1993.

[IMA93] IMA. Multimedia system services. Technical report, nteractive Multimedia Association, 1993. Version 2.0.

[ISO92] ISO-100744. Information processing – text and office systems – hypermedia/time-based structuring language — (HyTime). International Standard IS-100744, International Standards Organisation, Geneva, 1992.

[KAS94] Hermann Maurer Keith Andrews and Nick Scherbakov. Semantic modeling in hypermedia: An object-oriented approach. Technical report, Institute for Information and Computer Support New Media, Graz University of Technology, 1994.

[LHB93] G. van Rossan L. Hardman and D. C. A. Bulterman. Structured multimedia authoring. In *Proceedings of the first ACM Conference on Multimedia*, pages 283–289, 1993.

[Mac90] MacroMind. *Dynamic Media Authoring Tool for the Apple Macintosh Director Version 2.0.* Macromind, 1990.

[MG95] Multimedia and Hypermedia Expert Group. Information technology - coding of multimedia and hypermedia information- part 1. Technical report, ISO/IEC, 1995.

[NO94] P. Newhouse and R. Oliver. *Authorware 2: An Introduction to Interactive Multimedia with Authorware Professional 2.* 1994.

[Pei95] Mauro Sergio Oliveira Peixoto. Modelagem de sistemas de hipertexto: Paradigmas e uma aplicação prática real. Master's thesis, Departamento de InformáUniversidade Federal de Pernambuco, July 1995.

[ROK90] H. Harada Ryichu Ogawa and A. Kaneko. Scenario based hypermedia: A model and a system. pages 38–51, November 1990.

[Sap93] Peter Sapaty. A brief intoduction to the wavelanguage. Technical report, Universitat Karlsruhe, February 1993.

[SB95] Daniel Schwabe and Simone D. J. Barbosa. Navigation modelling in hypermedia applications, 1995.

[Sig95] J. Siglar. Multimedia authoring systems frequently asked question (faq). Technical report, The Internet, 1995.

[Sun95a] Sun. The hotjava browser: A white paper. Technical report, Sun Microsystems Computer Corporation, 1995.

[Sun95b] Sun. The java language specification. Technical report, Sun Microsystems Computer Corporation, 1995.

[VM93] M. Vazirgiannis and C. Mourlas. An object oriented model for interactive multimedia presentations. *The Computer Journal*, 36(1), 1993.

[Wan93] P. Wanyer. Inside quicktime. *Byte*, 16(13), 1993.

Complementing the space-time constraints paradigm with event facilities

José Tarcísio Franco de Camargo[1]
Léo Pini Magalhães[2]
Alberto Barbosa Raposo[3]

Department of Computer Engineering and Industrial Automation (DCA)
Faculty of Electrical Engineering (FEE)
State University of Campinas (UNICAMP)
P.O. Box: 6101
Fax: +55 0192 39-1395
13081-970 Campinas, SP, Brazil

[1]tarcisio@dca.fee.unicamp.br
[2]leopini@dca.fee.unicamp.br
[3]alberto@dca.fee.unicamp.br

Abstract. According to the space-time constraints paradigm for computer animation, the motion of an object in an animation or simulation will be described through a set of rules, that represents the "behavior" of the object, and some space and time constraints imposed by the animator. This means that, to evaluate the motion of an object from some point to another, the animator must provide a set of equations (physical laws, for example) and some boundary values (constraints). This is the reflex of the great effort that have been done on the search for realistic motion in computer animation. Indeed, the behavior of almost all characters in an animation or simulation can be represented by physical and mathematical tools, such as Newton and Euler laws and differential calculus. However, at this point, the question is *if I have many objects to animate how can I deal with the interactions that may occur among these objects?* The problem of modelling interactions among actors and the animator suggests some other approach. To solve this kind of problem we split the computer animation control problem in two parts: a local control problem and a global control problem. The local control defines the "behavior" of each character and is based on space-time constraints paradigm, while the global control is responsible for the interactions and is based on an event driven strategy. This strategy was implemented using concepts regarding to DEDS (**Discrete Event Dynamic System**) and ESM (**Extended State Machines**). The obtained results allow us to simulate a wide range of systems, and confirm that space-time constraints paradigm can be greatly improved considering an event driven strategy.

1 Introduction

The use of computers in modelling the animation and simulation of many types of systems is still a recent research field in computer science. However, in the search for realistic motion almost all works done in this line of inquiry have focused on the use of physical laws as the basic modelling tools for every kind of system that has to be represented.

Following this line Andrew Witkin and Michael Kass proposed, in **[Witkin (1988)]**, the space-time constraints paradigm. According to them, a physically-based approach for modelling the motion of a character may be specified by:

- The character's physical structure and additional resources.
- What the character has to do, for example "move from here to there"
- How the motion is performed, for example "move smoothly avoiding to waste energy"

The specification the motion af an object, in such a way that it achieves given tasks and performs these tasks in a realistic way has been an elusive goal for computer animators.

Newton and Euler laws are usually the basic tools for modelling the behavior of the character, while constraints on initial, final or intermediate positions and velocities directly encode the "goals" of

the motion. This implies the use of ordinary or partial differential equations for modelling the behavior and structure of the character, and boundary values (constraints) to instance them.

This paradigm has been shown to be a valuable approach for modelling the motion of a single character, but it suffers from computational complexity growth when we have to consider the interactions among characters in an animation. Even recently proposed revisions for this paradigm, such as those presented in **[Ngo (1993)]** and **[Liu (1994)**, do not solve quite well this problem.

As an alternative for solving the problem of controlling the interactions in an animation, we may consider to complement the space-time constraints paradigm with an event driven strategy. To do so, we suggest to split the model of an animation in several parts: local control blocks and a global control block.

Suppose a system to be simulated with many "actors". We can assign to each actor a local control block that contains the "behavioral rules" of the single actor. That is, the physical laws which control a single actor are embedded in its respective local control block. This suggests a tool, such as space-time constraints, for modelling the respective actor and its behavior. For instace, the mass distribution of the actor with respect to some coordinate system is given by its inertia tensor matrix; the motion rules of the actor may be described by Newton-Euler dynamic equations and they may compose a local control block.

Since systems we are talking about are composed by many actors there could be interactions among them and the animator. Then, the global control block is needed to suggest a logical approach for modelling these interactions. That is, when an event occurs, how must an actor respond? Which actors are affected by this event? This way, depending on what is "going on" in the system, the actors can behave in such a way that could be different if something else had occurred. This approach was also studied in **[Kalra (1992)]**, but not splitting the control in local and global blocks.

At next we introduce the initial step of our proposal, i.e. splitting the animation process in two classes of control blocks: the global control block and the local control blocks and the event strategy applied to the control.

2 Modelling tools

Consider, for instance, figure 1, that represents a system, where we have two actors (represented by their respective local control blocks) and a global controller, that is responsible for the interactions.

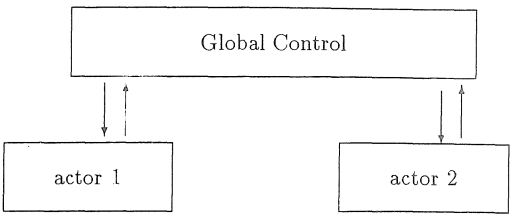


Figure 1: The flow of information among "the actors" in an animation.

According to figure 1 we can see that both actors in this system are connected to the global controller. The connections are made in such a way that each actor has an "outgoing channel" and an "incoming channel" with the global control block. Through these channels the global controller can send and receive any information regarding to the actors. We can also note that this process has feedback loops between the global control block and each actor. In addition, according to the last section, each actor in figure 1 has inside its local control block its behavioral rules and a logical structure that is responsible for the communication with the global controller. On the other hand the global control block contains only a logical structure that is responsible for the coordination of the actors.

Figure 1 may be an example for a wide range of systems, such as an assembly line or a video game, where the evolution of the system in time depends on the complex interactions of the timming of various discrete events, such as the arrival or departure of a job or the initiation and completion of a task or message. So, the state of such system changes at these discrete instants of time rather than continuously.

It is possible to model a system of this kind through the use of the space-time constraints paradigm. The problem is that for modelling the interactions that may occur we will need a large (and complex) set of equations.

This problem has driven us to model this kind of animation through the use of "space-time-event" constraints. This means that, when an event occurs in the system, or at any time instant, the global controller may send a proper command to an actor or it can receive any information from an actor about its status or an occurred event. This kind of system is usually called 'Discrete Event Dynamic System - DEDS" **[Ho (1989)]** and, to build a system of this kind, we can use several modelling tools, such as "Extended State Machines - ESM".

ESM features are suitable for modelling "Space-Time-Event" oriented systems and they have been frequently used in our work [**Camargo (1994)**]. At next we introduce the basic ESM definitions we use in our research.

2.1 ESM definitions

A basic ESM may be represented by a quintuple **(X, Y, C, L, A)** where, according to [**Ostroff (1989)**]:

X is a variable whose instace denotes the current state of the machine.

Y is a set of generic data variables that may be used by the ESM.

C is a set of communication channels. A communication channel can be considered as a one direction communication line connecting two ESMs, through which some data can be transmitted.

L is a set of event labels.

A is a set of basic actions.

Each basic action in **A** is given by the quadruple **(cs, guard, operation, ds)**, where:

cs is the current state of the ESM,

ds is the destination state of the ESM,

guard is a boolean-valued expression, and

operation is one of following three possible operations.

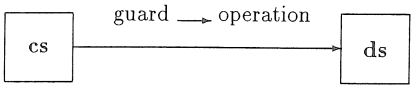
The three possible operations are given by:

1. An **assign operation** given by "α[y:a]" (this means: "a" is assigned to "y" on the occurrence of *alpha*) where "α" is an event label, "y" is a data variable and "a" is an expression. The expression "a" must be of the same type as the data variable "y". The event label may occur by itself in an operation, i.e. [y:a] is optional. Multiple simultaneous assignments in the same operation are denoted "$\alpha[y_1 : a_1, y_2 : a_2]$".

2. A **send operation** is given by "c!m", where "c" is a channel in **C** and "m" is a message. A message is either an expression or an event label.

3. A **receive operation** is given by "c?r", where "c" is a communication channel, and "r" is either a data variable or an event label.

As an example, the graph:



is a pictorial representation of an event. If the **guard** is left out, then it is assumed to be true. An informal interpretation of the above event is: "if the ESM is currently in the state **cs** and if the guard evaluates to true, then the edge is traversed while doing **operation**, after which the ESM will be in state **ds**"

Communicating operations can be used either to command the execution of a specific event or to communicate a message. An informal description of these two types of communication is:

COMMAND TO EXECUTE AN EVENT - The send operation "c!α" in some ESM M_1 is read: "send the command to do "α" over channel "c" (to some other ESM e. g. M_2)". The receive operation "c?α" in M_2 is read: "receive the command to do "α" (e. g. from M_1) over channel "c" and execute the command α ". The send and receive operations, when synchronized, result in the command and execution of the event "α".

COMMUNICATING A MESSAGE - Let "m" be an expression (the value of "m" is the message), and let "r" be a data variable. Then the operation "c!m" in some ESM M_1 is read: "send the message "m" over channel "c" (to some other ESM e. g. M_2)". The corresponding operation "c?r" in M_2 is read: "receive a message (from e. g. M_1) over channel "c" and put the message in "r" ". The send and receive operations togheter result in a distributed assigment of "m" to "r".

Note that, a send operation "c_1!m" matches a receive operation "c_2?r" if $c_1 = c_2$ and, either "m" and "r" are the same event label, or "r" is a data variable and "m" is an expression with the same type as "r". A send action from some ESM "M" cannot have a matching receive action in "M" as there is no internal communication within an ESM ifself (a channel represent a one-to-one connection from "M" to some other ESM).

The above definitions do not complete the entire description of ESMs, but they are sufficient for the purpose of this paper. Further details can be found in [**Ostroff (1989)**].

3 An Example: modelling a bipedal structure

We believe that the best way to explain our proposal is through an example. So, consider the bipedal "actor" shown in Figure 2. The figure presents an articulated human-like body that is composed by a set of articulated structures. When those structures are joined we have the bipedal figure. In order to simulate this articulated character we can consider two control levels, local and global. Each local control block is responsible for the effective displacement of its respective leg or arm, that is, a controller of this kind must control the motion of a single leg or arm from one place to another. There are several classical kinematic or dynamic techniques that can be applied to solve this local control problem, such as those presented in **[Korein (1982)]**, **[Craig (1989)]**, **[Goldenberg (1985)]**, **[McInnis (1986)]**, **[Morgan (1985)]**, **[Armstrong (1985)]**, **[Barzel (1988)]**, **[Witkin (1988)]**, **[Ngo (1993)]**, **[Liu (1994)]**, etc.

However, it is not enough to control only the motion of each leg individualy, since, for a bipedal structure, there must be coordination between the legs, otherwise we can not move the entire body. For example, both legs are not allowed to be out of the ground at the same time when the actor is walking. In this way, when dealing with the interaction between legs, we are facing a global control problem.

The solution for the this control problem can be found in DEDS formulation. Our global control approach is partially based on the model presented in **[Girard (1985)]**, and the final solution that we propose uses the same tools as those presented in **[Ho (1989)]**, **[Ostroff (1989)]** and **[Cohen (1989)]**.

Note that the scheme presented in fig 1 represents exactly the control problem that we are dealing.

For this structure, a **gait pattern** describes a sequence of lifting and placing of the feet. The pattern repeats itself as the figure moves: each repetition of the sequence is called the gait cycle. The time (or number of frames) taken to complete a single gait cycle is the period **P** of the cycle.

During each gait cycle period any given leg will spend a percentage of that time on the ground. This fraction is called the duty factor of the leg. In our example, the duty factor may be used to distinguish between the walking and running gaits of bipeds. In addition, we can define the stroke as the distance traveled by the body while one leg is still on the ground.

We can see each one of the legs as a basic ESM where, associated with each leg "i", we have the following states:

S_i The **support state**. We will call the time a leg spends on the ground its **support duration**. Its respective phase is the support state.

T_i The **transfer state**. We will call the time a leg spends in the air its **transfer duration**. Its respective state is the transfer state.

We still have associated with each leg "i" the following events:

α_i The leg is lifting the ground.

β_i The leg is placing on the ground.

The flow of information between the controller and each leg is done through the communication channels:

m_i Send information from the leg to the controller.

c_l Send information from the controller to the leg.

As a basic control specification we have:
It is not allowed to both legs be in the transfer state at the same time when the body is walking.

For each leg "i", (L_i), we can build the following ESM:

ESM L_i:
$L_i = [(x_i), \emptyset, (m_i, c_i), (\alpha_i, \beta_i), A_i]$
$(x_i) = (S_i, T_i)$

$$A_i = [\ [T_i, \beta_i, \emptyset, S_i]\ ,\ [S_i, c_i?\alpha_i, \emptyset, T_i]\ ,\ [S_i, TRUE, m_i!x_i, S_i]\ ,\ [T_i, TRUE, m_i!x_i, T_i]\ ]$$

The complete system must also consider the torso and the arms of the body. However, for the sake of simplicity, we will only treat the legs in this paper. Now, we will treat the global control problem under a simple point of view. In this case, the actor will move in a straight line indefinitely, and the controller may be represented as follow:

ESM Controller:
Controller =
$[(x_c), (y_l, y_r), (m_l, m_r, c_l, c_r), (\alpha_l, \alpha_r), A_c]$
$(x_c) = (L_1, L_2, L_3, L_4)$

$$A_c = [\ [L_1, m_l?y_l, \emptyset, L_2]\ ,\ [L_2, y_l = T_l, \emptyset, L_1]\ ,\ [L_2, y_l = S_l, c_r!\alpha_r, L_3]\ ,\ [L_3, m_r?y_r, \emptyset, L_4]\ ,\ [L_4, y_r = T_r, \emptyset, L_3]\ ,\ [L_4, y_r = S_r, c_l!\alpha_l, L_1]\ ]$$

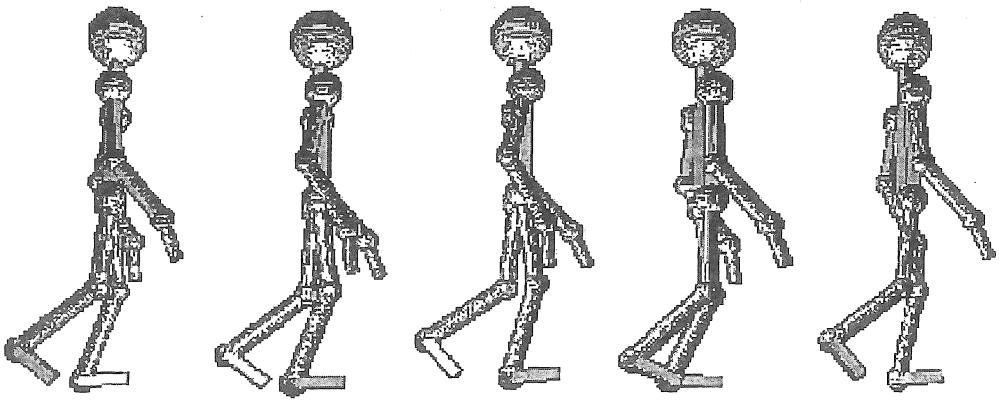


Figure 2: Frames generated for a bipedal structure simulation.

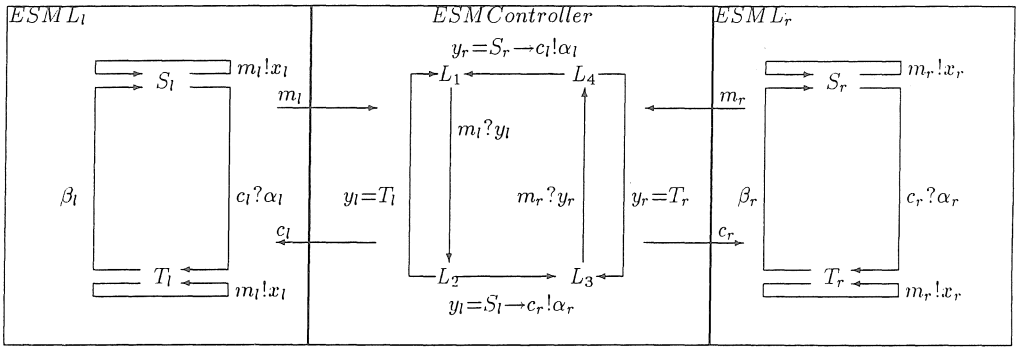


Figure 3: The system actors || controller.

A pictorial representation for the set actors - controller is shown in Figure 3.

4 Implementation

We have implemented a system called TOOKIMA (*a TOOl KIt for scripting computer Modeled Animation*) that defines a set of tools for the algorithmic description of animations. This system is able to describe the kinematic/dynamic motion of computer modelled objects.

The TOOKIMA system is, indeed, composed by three sub-systems:

Object Modelling Tool: With this sub-system we can model the objects that will take part in the animation process.

Visualization Tool: A Scanline based algorithm is used for the visualization of the frames generated by the system.

Scripting Tool: This sub-system is responsible for the integration of the two above. In addition, it provides a "language" for temporal synchronization and description of the scenes that take part in the animation. A script of an animation process is described in this pseudo-language that contains all information about the objects, the furniture and the actions that will compose the animation.

According to TOOKIMA's structure, we have built the ESMs of this example as "sub-routines" in the script of the animation. The script for the frames presented in figure 2 is:

```
/* Definition of ESMs and initial states */
DEF_ESM(CTRL,"L1")
DEF_ESM(LEFT_LEG,"SL")
DEF_ESM(RIGHT_LEG,"SR")

/* Definition of events */
```

```
DEF_EVENT(BL)
DEF_EVENT(AL)
DEF_EVENT(BR)
DEF_EVENT(AR)

/* Definition of channels */
DEF_CHANNEL(ML)
DEF_CHANNEL(MR)
DEF_CHANNEL(CL)
DEF_CHANNEL(CR)

TIME_INTERVAL(0.,11.)
SAMPLES(12)

/* External functions */
FUNCTION(move_right_leg)
FUNCTION(move_left_leg)

START_SIMULATION

/* ESM Controller */
BEGIN_ESM(CTRL)
CHANNEL(ML)
CHANNEL(MR)
CHANNEL(CL)
CHANNEL(CR)
ACTION(CTRL,"L1",MATCH(ML,"TL"),NO_TASK,"L1")
ACTION(CTRL,"L1",MATCH(ML,"SL"),CR="AR","L2")
ACTION(CTRL,"L2",MATCH(MR,"TR"),NO_TASK,"L2")
ACTION(CTRL,"L2",MATCH(MR,"SR"),CL="AL","L1")
END_ESM

/* ESM Left Leg */
BEGIN_ESM(LEFT_LEG)
CHANNEL(ML)
CHANNEL(CL)
EVENT(BL)
ACTION(LEFT_LEG,"SL",TRUE,ML=LEFT_LEG,"SL")
ACTION(LEFT_LEG,"SL",MATCH(CL,"AL"),
       move_left_leg(),"TL")
ACTION(LEFT_LEG,"TL",TRUE,ML=LEFT_LEG,"TL")
ACTION(LEFT_LEG,"TL",BL,NO_TASK,"SL")
END_ESM

/* ESM Right Leg */
BEGIN_ESM(RIGHT_LEG)
CHANNEL(CH1R)
CHANNEL(CH2R)
EVENT(BR)
ACTION(RIGHT_LEG,"SR",TRUE,MR=RIGHT_LEG,"SR")
ACTION(RIGHT_LEG,"SR",MATCH(CR,"AR"),
       move_right_leg(),"TR")
ACTION(RIGHT_LEG,"TR",TRUE,MR=RIGHT_LEG,"TR")
ACTION(RIGHT_LEG,"TR",BR,NO_TASK,"SR")
END_ESM

END_SIMULATION
```

At the top of the simulation file we define the ESMs, the events, and the channels that will take part in the animation. At next, we have the timming definitions of the animation and the definitions of the external functions. These external functions contain the behavioral rules of the motion of each leg. This may be any set of kinematic or dynamic equations. Finally, we have the set of possible actions that an actor may perform.

As can be seen in this script, the global control block is composed only by a set of logical expressions used to manage the interactions among the other blocks. On the other hand, each local control block has a set of logical expressions to interact with the controller and, also, a function that will perform the efective displacement of each leg, according to a space-time constraints approach.

5 Conclusions

"Space-Time Constraints" paradigm is, today, the most commom modelling tool for computer animation. It is frequently used for modelling many kind of systems around us. However, it is not suitable when considering the presence of discrete events in the system, or even considering a large set of possible interactions among the actors in an animation.

Any animation process can be thought as a sequence of discrete events where the "actors" in the system are changing their states according to the occurrence of the events. Thinking this way, we can model animations as discrete event dynamic systems (DEDS). A quite commom tool for modelling DEDS is the extended state machine (ESM) that appears to be suitable for applications in computer modeled animation or simulation.

Following this line of inquire, we have considered in this paper an "extended" paradigm for computer modeled animation. "Space-Time-Event Constraints" paradigm appers to cover a lot more than "Space-Time". Also, the use of local and global controllers are suitable for the implementation of a wide range of animation models. If we consider the global and local controllers as ESMs, it allows us to split an initial complex problem in several smaller problems.

In addition, the global control structure let us free to think about the local control blocks. That is, no matter which local control technique we adopt, the global control block does not change. This way, when solving the local control problem we can choose among several techniques such as a kinematic or a

dynamic model.

6 References

[**Armstrong (1985)**] W. W. Armstrong and M. W. Green, *The dynamics of articulated rigid bodies for purposes of animation*, The Visual Computer, **1**,4,231-240 (1985)

[**Barzel (1998)**] R. Barzel and A. H. Barr, *A modeling system based on dynamic constraints*, Computer Graphics, **22**,4,179-188 (1988)

[**Brotman (1988)**] L. S. Brotman and A. N. Netravali, *Motion interpolation by optimal control*, Computer Graphics, **22**,4,309-315 (1988)

[**Camargo (1994)**] J. T. F. de Camargo, L. P. Magalhães and A. B. Raposo, *Modeling motion simulation with DEDS*, Proc. of 13^{th} IFIP World Computer Congress-94, **2**,162-167 (1994)

[**Cohen (1989)**] G. Cohen, P. Moller, J. Quadrat and M. Viot, *Algebraic tools for the performance evaluation of discrete event systems*, Proc. IEEE, **77**,1, 39-57 (1989)

[**Craig (1989)**] J. J. Craig, *Introduction to robotics: mechanics and control*, 2^{nd} ed, Addison-Wesley Publishing Company (1989)

[**Girard (1985)**] M. Girard and A. A. Maciejewski, *Computational modeling for the computer animation of legged figures*, Computer Graphics, **19**,3,263-270 (1985)

[**Goldenberg (1985)**] A. A. Goldenberg, B. Benhabib and R. G. Fenton, *A complete generalized solution to the inverse kinematics of robots*, IEEE Journal of Robotics and Automation, **RA-1**,2,14-20 (1985)

[**Ho (1989)**] Y. Ho, *Dynamics of discrete event systems*, Proc. IEEE, **77**,1, 3-6 (1989)

[**Kalra (1992)**] D. Kalra and A. H. Barr, *Modelling with time and events in computer animation*, Proceedings of EUROGRAPHICS-92, **11**,3,35-58, (1992)

[**Liu (1994)**] Z. Liu, S. J. Gortler and M. F. Cohen, *Hierarchical spacetime control*, Computer Graphics Proceedings, Annual Conference Series, 35-42, (1994)

[**Korein (1982)**] J. U. Korein and N. I. Badler, *Techniques for generating the goal-directed motion of articulated structures*, IEEE Computer Graphics and Applications, 71-81, November (1982)

[**McInnis (1986)**] B. C. McInnis and C. F. Liu, *Kinematics and dynamics in robotics: a tutorial based upon classical concepts of vectorial mechanics*, IEEE Journal of Robotics and Automation, **RA-2**,4,181-187, (1986)

[**Morgan (1985)**] R. G. Morgan and U. Ozguner, *A descentralizated variable structure control algorithm for robotic manipulators*, IEEE Journal of Robotics and Automation, **RA-1**,1,57-65 (1985)

[**Ngo (1993)**] J. T. Ngo, *Spacetime constraints revised*, Computer Graphics Proceedings, Annual Conference Series, 343-350 (1993)

[**Ostroff (1989)**] J. S. Ostroff, *Temporal logic for real time systems*, Research Studies Press Ltd., John Wiley & Sons Inc. (1989)

[**Thalmann (1985)**] N. M. Thalmann and D. Thalmann, *Computer Animation - Theory and Practice*, Springer-Verlag (1985)

[**Witkin (1988)**] A. Witkin and M. Kass, *Spacetime constraints*, Computer Graphics, **22**,4,159-168 (1988)

Uso de Deducciones de no Viabilidad en el Cálculo de Arco-Consistencia

Frank D. Valencia.
e-mail: fvalenci@sofia.univalle.edu.co

Camilo Rueda.
e-mail: crueda@borabora.univalle.edu.co

Grupo AVISPA
Pontificia Universidad Javeriana de Cali
Dirección: Calle 18 Num. 118-250 Vía Pance
Teléfono: (57) 2-5552175
Fax: (57) 2-5552823
Cali, Colombia.

Resumen

Los problemas de Satisfacción de Restricciones, conocidos como CSP ("Constraint Satisfaction Problems") constituyen actualmente una de las áreas centrales de la Inteligencia Artificial. Las técnicas de Arco-Consistencia (AC) se utilizan ampliamente en el tratamiento de CSP. Recientemente se han propuesto para el cálculo de Arco-Consistencia los algoritmos AC6++ y AC-7, que mantienen las denominadas cuatro propiedades deseables de un algoritmo de Arco-Consistencia y con base en ellas se afirma su optimalidad. Se muestra en este artículo que dichas propiedades pueden no ser suficientes. Se presenta una nueva propiedad deseable, independiente de la semántica de las restricciones, no considerada anteriormente por ningún algoritmo de AC. Esta propiedad es *necesaria* para garantizar algoritmos óptimos. Su efectividad se muestra analítica y experimentalmente modificando AC6++ y AC-7.

1. INTRODUCCIÓN

El concepto de problema de satisfacción de restricciones (CSP) constituye hoy un área fundamental de la inteligencia artificial. Esta importancia se debe a que bajo este concepto pueden formularse muchos problemas interesantes de carácter combinatorio. Un CSP se define por un conjunto de variables sujetas a un conjunto de restricciones y un conjunto finito (dominio) de valores asociado a cada variable. Una solución a un CSP, es una asignación de valores a variables que satisfacen todas las restricciones.

Las técnicas de Consistencia, especialmente las de Arco-Consistencia (AC) han sido ampliamente estudiadas para simplificar el CSP antes o durante la búsqueda de soluciones. Sabin [7] ha mostrado experimentalmente la importancia de usar AC durante la búsqueda de soluciones.

Varios algoritmos de AC han sido propuestos, comenzando en 1972 con el algoritmo de Waltz [11] y terminando recientemente con los óptimos AC-6++ [3] y AC-7 [2]. Entre los algoritmos de AC, son de especial interés: AC-3 [5], AC-4 [6], AC-5 [9], AC-6 [1], AC-*Inference* [4], AC-6++ y AC-7. AC-3 es el modelo de los restantes. AC-4 introduce estructuras de datos para un procesamiento dinámico. AC-5 es un algoritmo genérico que puede ser optimado para ciertas clases importantes de restricciones, o especializado para obtener AC-3 o AC-4. AC-6 es un algoritmo incremental basado en AC-4. AC-*Inference* es un algoritmo que al igual que AC-5 puede tomar ventajas de las propiedades de las restricciones. AC-6++ y AC-7 refinan AC-6, manteniendo *las cuatro propiedades deseables* de un algoritmo de Arco-Consistencia.

El estándar de comparación en los algoritmos de AC lo constituyen el número de chequeos de restricciones realizados. Con base en este estándar, tres de las cuatro propiedades deseables establecen condiciones para reducir el número de chequeos de restricciones. La cuarta propiedad establece una condición sobre la complejidad en espacio.

En este trabajo se presenta una nueva propiedad deseable, no considerada por ningún algoritmo de AC previo, que ofrece una reducción en el número de chequeos de restricciones, como se muestra analítica y experimentalmente al modificar AC-7 y AC6++. Estas modificaciones serán denominadas AC6-3+ y AC-7+, respectivamente.

Este artículo está organizado así: En la sección 2 se presentan definiciones generales; en la sección 3 se tratan las cuatro propiedades deseables; en la sección 4 se presenta la nueva propiedad deseable, una descripción genérica de un algoritmo que la considera y su análisis; en la sección 5, se muestran resultados experimentales. Las conclusiones se presentan en la sección 6.

2. PRELIMINARES

Un CSP se define como una tripla $<V,D,C>$. $V=\{x_1,...,x_n\}$ es el conjunto de variables, con su domino de valores asociado en $D=\{D_1,...,D_n\}$, donde el dominio de la variable x_i es D_i. C es un conjunto de restricciones $R_{(i_1,...,i_u)}$ que denota combinaciones de valores permitidas para las variables $x_{i_1},...,x_{i_u}$ de V, (i.e., $R_{(i_1,...,i_u)} \subseteq D_{i_1} \times ... \times D_{i_u}$). Utilizamos el predicado $R_{(i_1,...,i_u)}(v_{i_1},...,v_{i_u}) \Leftrightarrow (v_{i_1},...,v_{i_u}) \in R_{(i_1,...,i_u)}$. Se dice que un CSP es *binario* si todas las restricciones en C son binarias, es decir, de la forma $R_{(i,j)}$. Por claridad en este trabajo se trata el caso binario, sin embargo la generalización a n-ario es inmediata.

Calcular Arco-Consistencia significa eliminar algunos valores de los dominios, que no pueden aparecer en ninguna solución. Estos valores se dicen no *viables*.

Definición 1. $w \in D_j$ es un *soporte* para $v \in D_i$ en $R_{(i,j)}$ ssí $R_{(i,j)}(v,w)$. $v \in D_i$ es *viable con respecto a* $R_{(i,j)}$ ssí $\exists w \in D_j$: w es un soporte para v en $R_{(i,j)}$. $v \in D_i$ es *viable* ssí $\forall R_{(i,j)} \in C$: v es *viable con respecto a* $R_{(i,j)}$.

Los algoritmos de AC operan sobre un grafo de restricciones que representa al CSP. Los nodos de este grafo corresponden a las variables y los arcos a las restricciones. Por cada restricción $R_{(i,j)}$, el grafo tiene dos arcos (i,j) y (j,i). El arco (i,j) es asociado a $R_{(i,j)}$ y el arco (j,i) a $R_{(j,i)}$, que es igual $R_{(i,j)}$ excepto que sus argumentos están intercambiados (i.e., $R_{(j,i)}(w,v)=R_{(i,j)}(v,w)$). Se dice que $R_{(i,j)}$ y $R_{(j,i)}$ cumplen *la propiedad de bidireccionalidad.* Sin pérdida de generalidad, se asume que sobre un mismo par de variables existe a lo sumo una restricción. Entonces, Un CSP binario $<V,D,C>$ se representa con el grafo de restricciones $G=<V,A>$ donde $A=\{ (x_i,x_j) \mid R_{(i,j)} \in C \vee R_{(j,i)} \in C\}$. En este trabajo cada arco (x_i,x_j) en A se abrevia como (i,j). Denotamos $Arcos(G)=A$, y $Nodos(G)=V$.

La siguientes definiciones de uso general describen Arco-Consistencia en un arco y en un grafo [9]:

Definición 2. Sea $(i,j) \in Arcos(G)$. El arco (i,j) es *Arco-Consistente con respecto a* D_i y D_j ssí $\forall v \in D_i : v$ es viable con respecto a $R_{(i,j)}$. Sea $\rho' = D_1 \times ... \times D_n$. G es *Arco-Consistente* con *respecto a* ρ' ssí $\forall (i,j) \in Arcos(G) : (i,j)$ es Arco-Consistente con respecto a D_i y D_j. Sea $\rho = D_1 \times ... \times D_n$. Sea $\rho' \subseteq \rho$. G *es maximalmente arco-consistente con respecto a* ρ' *en* ρ ssí G es Arco-Consistente con respecto ρ' y no existe otro ρ'' con $\rho' \subset \rho''$ tal que G es Arco-Consistente con respecto ρ''.

Dado G y ρ, el propósito de un algoritmo de AC es calcular un ρ' tal que G sea maximalmente arco-consistente con respecto ρ' en ρ.

En el cálculo de Arco-Consistencia, se define un ordenamiento total "$\leq$" en los dominios.

Definición 3. El soporte inferior en D_j para un valor $v \in D_i$, es un valor $SI_{[(i,j),v]}$ tal que cualquier valor en D_j menor que $SI_{[(i,j),v]}$ no es compatible con v. La *propiedad del soporte inferior* establece: $\forall (i,j) \in Arcos(G) \left(\forall v \in D_i, \forall w \in D_j \left(w < SI_{[(i,j),v]} \Rightarrow \neg R_{(i,j)}(v,w) \right) \right)$. Note que $SI_{[(i,j),v]}$ no necesariamente es un soporte.

En las próxima sección se mostrará que las propiedades del soporte inferior y de bidireccionalidad, son bastante útiles en el Cálculo de Arco-Consistencia.

3. LAS CUATRO PROPIEDADES DESEABLES

El estado del arte en el cálculo de Arco-Consistencia reside en cuatro *propiedades deseables* que definen el comportamiento ideal de un algoritmo de AC [3]. Estas propiedades están orientadas a determinar un número de chequeos óptimo con una complejidad espacial práctica y pueden ser derivadas de las propiedades del soporte inferior y de la bidireccionalidad de las restricciones.

Las propiedades deseables de un algoritmo de arco-consistencia establecen que:

1. Nunca se debe chequear $R_{(i,j)}(v,w)$ si existe un w' en D_j tal que $R_{(i,j)}(v,w')$.
2. Nunca se debe chequear $R_{(i,j)}(v,w)$ si existe un w' en D_j tal que $R_{(j,i)}(w',v)$.
3. Nunca se debe chequear $R_{(i,j)}(v,w)$ si:
   a. $R_{(i,j)}(v,w)$ ha sido chequeada previamente,
   o b. $R_{(i,j)}(v,w)$ ha sido chequeada antes.

4. Se debe tener una complejidad espacial $O(ed)$, donde $e=|C|$ y $d=\underset{1\le i\le n}{Máx}(|D_i|)$.

La propiedad 1, se puede mantener utilizando la propiedad del soporte inferior. La propiedad 2, utilizando soporte inferior y bidireccionalidad. La propiedad 3a, utilizando soporte inferior, la 3b utilizando soporte inferior y bidireccionalidad. La propiedad 4 es una consideración práctica sobre el tamaño de las estructuras de datos. AC-3 no mantiene las propiedades 1,2,3. AC-4 no mantiene 1,2,3b,4. AC-6 no mantiene 2 y 3b (las propiedades derivadas de la bidireccionalidad). AC-*Inference* no mantiene la propiedad 4. AC6++ y AC-7 mantienen la cuatros propiedades, y por esta razón son considerados óptimos en [3].

AC6++ y AC-7 solo difieren en la forma en que los valores se tratan. Mientras AC6++ se aproxima en esta forma a AC-6, AC-7 coincide en esto con AC-*Inference*. La idea de AC-7 (y de AC-*Inference*) consiste en propagar los efectos de eliminar un valor no viable (i.e., reconsiderar la viabilidad de los valores que éste soportaba) tan pronto como sea posible, con el fin de reducir chequeos de restricciones innecesarios y también de descubrir dominios vacíos rápidamente. En la práctica, en promedio, AC-7 ha demostrado ser superior a AC6++ al utilizar ésta idea.

En [2] se afirma que con estas propiedades AC6++ garantiza un número de chequeos óptimo dado un ordenamiento de valores, de variables y de arcos (el orden en que los valores, variables y arcos son considerados para calcular Arco-Consistencia). Sin embargo es posible encontrar casos excepcionales en los cuales AC-6 puede chequear considerablemente menos restricciones que AC6++. Son casos en los que es imposible establecer el mismo ordenamiento de valores en AC-6 y AC6++ [8].

Las anteriores propiedades son también importantes como heurísticos para reducir el chequeo de restricciones, para garantizar que un chequeo no sea realizado más de una vez, y para no realizar chequeos innecesarios.

4. LA NUEVA PROPIEDAD DESEABLE

La eliminación de un valor de un dominio impone la reconsideración de la viabilidad de valores en otros dominios para los cuales éste era soporte. Siguiendo la idea de AC-7, si se desea propagar las

consecuencias de la eliminación de un valor tan pronto ésta como suceda, es razonable pensar que se deberían realizar estas eliminaciones tan pronto como sea posible.

La nueva propiedad deseable ayuda a encontrar valores no viables rápidamente, con base en el principio que denominamos deducciones *de no viabilidad.* Tales deducciones evitan, por una parte, que se busquen soportes de valores no viables y, por otra, que valores no viables se asignen como soportes de otros. Esta propiedad no es considerada por ningún algoritmo previo.

La quinta propiedad deseable establece que valores para los cuales se pueda deducir que no son viables no deben ser chequeados. Esta propiedad puede expresarse de la siguiente manera:

5. Nunca se debe chequear $R_{(i,j)}(v,w)$ cuando se pueda deducir que v o w no son viables.

AC-5 deduce la no viabilidad de un valor en clases especiales de restricciones, pero no en restricciones arbitrarias.

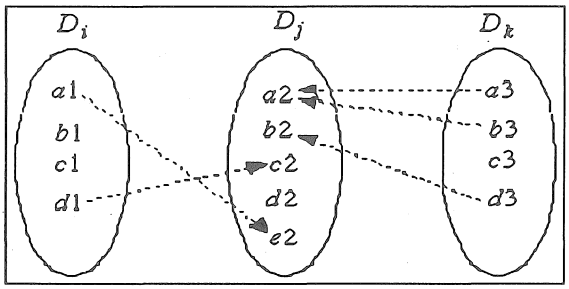


Figura 1. Una flecha de x a y, indica que y es el primer soporte de x.

Suponga que algún algoritmo de AC (usando un ordenamiento total en los dominios) ha chequeado todos los valores en D_i buscando soporte en D_j (ver Figura 1). Los valores $a1$ y $d1$ tienen a $e2$ y $c2$, respectivamente, como primer soporte en D_j. Los valores $b1$ y $c1$ al no tener soporte en D_j se eliminan. Seguidamente, todos los valores D_k se chequean buscando un soporte en D_j. El valor $c3$ es eliminado por no tener soporte en D_j, pero $a3$ y $b3$ tienen como soporte a $a2$, y $d3$ tiene a $b2$ como soporte. Sin embargo, se sabe que los valores $a2$ y $b2$ no son viables, pues todos los valores de D_i fueron chequeados y ninguno fue compatible con ellos (por el concepto de primer soporte y bidireccionalidad). No eliminar $a2$ y $b2$ cuando es posible deducir su no

viabilidad, puede producir propagaciones y chequeos innecesarios. No obstante, ninguno de los algoritmos previos evitan subsecuentes chequeos con $a2$ y $b2$.

Las cuatro propiedades deseables evitan chequear una pareja de valores cuando ésta ha sido anteriormente chequeada. La nueva propiedad propuesta evita chequear una pareja de valores cuando alguno de sus componentes se deduce no viable. Ahora se presentarán dos formas de deducciones de no viabilidad que son fácilmente adaptables en AC6++ y AC-7.

4.1 Propiedad 1 (Mínimo Soporte Inferior). Sea $MSI_{(i,j)} = \underset{v \in D_i}{Min}(SI_{[(i,j),v]})$ entonces $\forall w \in D_j \ (w < MSI_{(i,j)} \Rightarrow w \text{ no es viable})$.

Demostración : La propiedad del soporte inferior es equivalente a:

$\forall v \in D_i, \forall w \in D_j \ w < SI_{[(i,j),v]} \Rightarrow \forall v \in D_i, \forall w \in D_j : \neg R_{(i,j)}(v,w)$ (E.1).

Trivialmente, $\forall w \in D_j (\ w < MSI_{(i,j)} \Rightarrow \forall v \in D_i \ (w < SI_{[(i,j),v]}))$, que es equivalente a:

$\forall w \in D_j (\ w < MSI_{(i,j)}) \Rightarrow \forall w \in D_j, \forall v \in D_i \ (w < SI_{[(i,j),v]})$ (E.2).

De (E.2) y (E.1) se deduce: $\forall w \in D_j \ (w < MSI_{(i,j)} \Rightarrow \forall v \in D_i : \neg R_{(i,j)}(v,w))$. Por bidireccionalidad se tiene: $\forall w \in D_j (\ w < MSI_{(i,j)} \Rightarrow \forall v \in D_i : \neg R_{(j,i)}(w,v))$.

La propiedad establece que todos los valores en D_j menores que el mínimo soporte inferior de los valores en D_i, no son viables y por lo tanto pueden ser eliminados sin chequeos adicionales.

4.2 Propiedad 2 (Cero-Cardinalidad). Sea $PC_{[(i,j),v]} \geq \left| \{ v \in D_i | R_{(i,j)}(v,w) \} \right|$ una cota superior del número de soportes de $v \in D_i$ en D_j. Entonces $PC_{[(i,j),v]} = 0 \Rightarrow v$ no es viable.

4.3 Algoritmo. Aquí presentaremos un algoritmo genérico basado en AC-5 que considera las propiedades de Cero-Cardinalidad y la del Mínimo Soporte Inferior. Este algoritmo puede especializarse para obtener cualquiera de los algoritmos de AC.

Todos los algoritmos de AC usan una lista de espera Q. En este esquema genérico Q tiene elementos de la forma $<(i,j),w>$ donde (i,j) es un arco y w es un valor que ha sido borrado de D_j y justifica la necesidad de reconsiderar para Arco-Consistencia el arco (i,j) (i.e., para la propagación de los efectos de la eliminación de w). El procedimiento Encolar(i,Δ,Q) se usa cuando un conjunto Δ ha sido eliminado de D_i (Figura 2), para agregar a la cola Q, elementos de la forma

$<(k,i),v>$ donde (k,i) es un arco y $v \in \Delta$. En las especificaciones, un parámetro indexado con 0 (i.e., de la forma p_o) representa su valor en el momento de la invocación del procedimiento o función. El procedimiento Decolar (Q,i,j,w) saca un elemento $<(i,j),w>$ de la cola Q (Figura 2).

proc EnColar (**ent** i,Δ, **ent−sal** Q)
Pre: $\Delta \subseteq D_i \wedge x_i \in Nodos(G)$
Post: $Q = Qo \cup \{<(k,i),v> | (k,i) \in Arcos(G) \wedge v \in \Delta\}$.

proc DeColar (**ent−sal** Q, **sal** i,j,w)
Post: $<(i,j),w> \in Qo \wedge Q = Qo - \{<(i,j),w>\}$.

Figura 2. Operaciones sobre Colas.

proc ArcCons (**ent** i,j **sal**, Δ_i, Δ_j)
Pre: $(i,j) \in Arcos(G)$.
Post: $\Delta_i = \{v \in D_i | \forall w \in D_j : \neg R_{(i,j)}(v,w)\} \wedge$
$\Delta_j = \{w \in D_j | w < MSI_{(i,j)}\}$.

proc LocalArcCons (**ent** i,j,w **sal** Δ_i, Δ_j)
Pre: $(i,j) \in Arcos(G) \wedge w \notin D_j$.
Post:
$\Delta_i = \{v \in D_i | R_{(i,j)}(v,w) \wedge \forall w' \in D_j : \neg R_{(i,j)}(v,w')\}$
$\wedge \Delta_j = \{w \in D_j | w < MSI_{(i,j)}\}$.

proc ArcCons (**ent** i,j **sal**, Δ_i, Δ_j)
Pre: $(i,j) \in Arcos(G)$.
Post: $\Delta_i = \{v \in D_i | \forall w \in D_j : \neg R_{(i,j)}(v,w)\} \wedge$
$\Delta_j = \{w \in D_j | PC_{[(j,i),w]} = 0\}$.

proc LocalArcCons (**ent** i,j,w **sal** Δ_i, Δ_j)
Pre: $(i,j) \in Arcos(G) \wedge w \notin D_j$.
Post:
$\Delta_i = \{v \in D_i | R_{(i,j)}(v,w) \wedge \forall w' \in D_j : \neg R_{(i,j)}(v,w')\}$
$\wedge \Delta_j = \{w \in D_j | PC_{[(j,i),w]} = 0\}$.

Figura 3. Procedimientos ArcCons y LocalArcCons modificados para considerar la propiedad del Mínimo Soporte Inferior (izq.) y la propiedad Cero-Cardinalidad (Der.).

El procedimiento ArcCons(i,j,Δ_i,Δ_j) calcula el conjunto de valores $\Delta_i \subseteq D_i$ para la variable x_i que no tienen soporte en D_j y *deduce* el conjunto de valores $\Delta_j \subseteq D_j$ que no tienen soporte en D_i. El procedimiento LocalArcCons(i,j,w,Δ_i,Δ_j) se usa para calcular el conjunto de valores $\Delta_i \subseteq D_i$, que ya no tienen soporte en D_j por causa de la eliminación de w de D_j y *deduce* también el conjunto de valores $\Delta_j \subseteq D_j$ que ya no tienen soporte en D_i. En la Figura 3, se especifican éstos procedimientos, usando como deducción la propiedad del mínimo soporte inferior (Izq.) y la propiedad Cero-Cardinalidad (Der.). En ningún algoritmo propuesto anteriormente se contempla Δ_j.

El algoritmo genérico tiene dos fases (ver Figura 4). En la fase de inicialización (líneas 2-7), a cada arco se le aplica Arco-Consistencia. En la fase de propagación (líneas 8-14), se propagan los efectos de las eliminaciones de valores no viables. Todo valor eliminado se guarda en la lista de

espera Q, hasta que se propaguen los efectos de su eliminación. El algoritmo termina cuando no existen más elementos en Q .

```
Algoritmo AC.
```
Post: Sea $\rho_o = D_{1_o} \times ... \times D_{n_o} \wedge \rho_f = D_{1_f} \times ... \times D_{n_f}$.
G es máximalmente Arco-Consistente con respecto a ρ_f en ρ_o.

```
{
1  Q←∅;
2  Para cada (i,j)∈Arcos(G) haga
3     ArcCons(i,j,Δi,Δj);
4     Encolar(i,Δi,Q);
5     Di←Di−Δi;
6     Encolar(j,Δj,Q);
7     Dj←Dj−Δj;
8  Mientras Q≠∅ haga
9     Decolar(Q,i,j,w);
10    LocalArcCons(i,j,w,Δi,Δj);
11    Encolar(i,Δi,Q);
12    Di←Di−Δi;
13    Encolar(j,Δj,Q);
14    Dj←Dj−Δj;
}
```

Figura 4. Algoritmo genérico de Arco-Consistencia.

Modificamos AC6++ añadiendo la propiedad del soporte inferior. Denominamos AC6-3+ el algoritmo resultante. Después de calcular el conjunto $\Delta_i \subseteq D_i$ mencionado anteriormente, se calcula el Mínimo Primer Soporte (i.e., $MSI_{(i,j)}$) y se eliminan los valores previos a éste.

Modificamos también AC-7 añadiendo la propiedad Cero-Cardinalidad. El algoritmo resultante lo denominamos AC-7+. Para usar la propiedad Cero-Cardinalidad, inicialmente cada $PC_{[(i,j),v]}$ se iguala a $|D_j|$. $PC_{[(i,j),v]}$ se decrementa cuando una restricción $R_{(i,j)}(v,w)$ no es satisfecha, cuando un soporte w' para v es eliminado, o cuando se encuentra que alguno de los valores para los cuales v es asignado como soporte, es eliminado. Cuando $PC_{[(i,j),v]}=0$, v se elimina de D_i .

4.3.1 Análisis.

Aquí se presentan un esbozo de la prueba de correctitud del algoritmo genérico AC.

Sea $\rho_o = D_{1_o} \times ... \times D_{n_o}$ y $\rho_f = D_{1_f} \times ... \times D_{n_f}$ tal que G es maximalmente Arco-Consistente con respecto a ρ_f en ρ_o. ρ_f es el conjunto que se espera calcular. Sea D_{i_k} $(1 \le i \le n)$ el dominio actual de la variable x_i después de la eliminación de un conjunto de valores de $D_{j_{k-1}}$ $(1 \le j \le n)$ (i.e., de la $k-$ésima ejecución de una eliminación de valores). Si $k=0$, D_{i_k} es el dominio inicial de la variable x_i cuando el algoritmo comienza. Sea $\rho_k = D_{1_k} \times ... \times D_{n_k}$. (i,a) denota el valor $a \in D_i$.

Todo valor (i,a) es eliminado de D_{i_k} sólo si no es viable, sea por el mismo principio de los algoritmos previos, sea por las propiedades Cero-Cardinalidad y Mínimo Soporte Inferior. Si cada valor (j,b) eliminado previamente no está en D_{j_f}, entonces (i,a) no está en D_{i_f}. Por definición $\rho_f \subseteq \rho_o$ y $\rho_k \subseteq \rho_{k-1}$, luego $\rho_f \subseteq \rho_k$. Entonces el algoritmo mantiene el invariante $\rho_f \subseteq \rho_k \subseteq \rho_{k-1} \subseteq ... \subseteq \rho_o$. Cada vez que un valor (j,b) es eliminado, es puesto en Q hasta que los valores que éste era soporte busquen nuevos soportes. Cada vez que se encuentra un valor (i,a) sin soporte, es eliminado de su dominio actual. Por lo tanto después de la fase de inicialización, cada valor tiene un soporte en los dominios actuales o en $Q' = \{w | < (i,k), w) \in Q\}$. Entonces el algoritmo mantiene el invariante: $\forall (i,j) \in Arcos(G) (\forall v \in D_{i_k} (\exists w \in (D_{j_k} \cup Q') : R_{(i,j)}(v,w)))$. El algoritmo termina cuando $Q = \varnothing$, entonces por definición, G es arco-consistente con respecto a ρ_k en ρ_o y del primer invariante se tiene que $\rho_f \subseteq \rho_k$, luego G es maximalmente arco-consistente con respecto a ρ_k en ρ_o (i.e. $\rho_f = \rho_k$) cuando el algoritmo termina.

Siendo extensiones de AC6++ y AC-7, los algortitmos AC6-3+ y AC-7+ conservan las cuatro propiedades deseables. Mantienen además la nueva propiedad deseable, por las propiedades del Mínimo Soporte Inferior y Cero-Cardinalidad. Dado un mismo ordenamiento (de valores, variables y arcos), todo chequeo realizado por AC6-3+ es realizado por AC6++ y todo chequeo realizado por AC-7+ es realizado también por AC-7, sin embargo AC6-3+ y AC-7 evitan chequeos de valores deducidos como no viables, que no necesariamente son evitados por AC6++ y AC-7.

5. RESULTADOS EXPERIMENTALES

En esta sección se muestran algunos resultados de los experimentos realizados al comparar los rendimientos de AC6++ vs. AC6-3+ y AC-7 vs. AC-7+. En los experimentos se contó el número

de chequeos de restricciones realizados por cada algoritmo. Cada comparación se realizó con cincuenta instancias de cada problema.

En el problema ZEBRA [10] se obtuvo los siguientes resultados: AC6++: 717 chequeos y AC6-3+: 636 chequeos, AC-7: 640 chequeos y AC-7+: 594 chequeos. En el problema de la teoría combinatoria sugerido en [10], se obtuvo: AC6++ : 977 chequeos y AC6-3+: 783 chequeos, AC-7 : 966 chequeos y AC-7+: 826 chequeos.

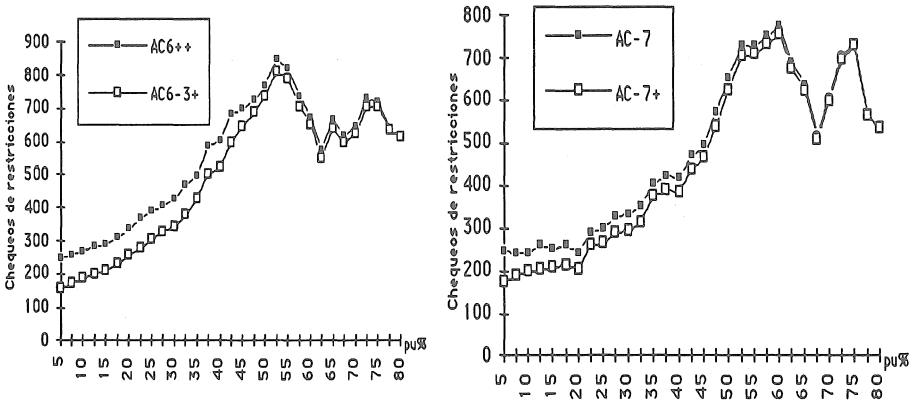


Figura 5. AC6++ vs. AC6-3+ (Izq.) y AC-7 vs. AC-7+ (Der.) en problemas aleatorios con n=20, d= 5, pc=30% y $0.05 \leq pu \leq 0.8$.

En los problemas aleatorios [1], n es el número de variables, d es el tamaño del dominio más grande, pc la probabilidad de que exista una restricción entre cualquier par de variables, y pu la probabilidad de que un par de valores cumplan una restricción. En la Figura 5, se muestran algunos resultados. En promedio, el rendimiento de AC6-3+ y AC-7+ es alrededor del 10% superior al de AC6++ y AC-7, respectivamente. Además, se observó que el número de valores deducidos como no viables fue proporcional al porcentaje de mejora. Aún cuando el número de valores deducidos como no viables fue pequeño, el número de chequeos fue apreciablemente reducido.

6. CONCLUSIONES

Se definió una nueva propiedad deseable para reducir el número de chequeos de restricciones realizados por un algoritmo de AC. Para considerar esta propiedad deseable, se definieron dos propiedades de deducción de no viabilidad. Como esta propiedad depende de la no viabilidad de los

valores, es muy práctica en problemas con fuertes propiedades estructurales (i.e., muchas restricciones, restricciones fuertes, muchas variables, dominios grandes, etc.). Estas propiedades son fácilmente adaptables a los dos mejores algoritmos de AC previamente conocidos, mejorando apreciablemente su comportamiento. Trabajos futuros en cálculo de Arco-Consistencia deben considerar esta nueva propiedad deseable.

REFERENCIAS

[1] Bessière, C. and Cordier, M. Arc Consistency and Arc Consistency again. In: Artificial Intelligence. Vol. 65 (1994) p. 179-190.

[2] Bessière, C. ; Frueder E.C. and Regin J.C. Using Inference to Reduce Arc Consistency Computation. In: *Proc.* ECAI'94.

[3] Bessière, C., Regin, J.C. Using bidirectionality to speed up arc-consistency processing. In: *Proc.* ECAI'94.

[4] Freuder E.C. Using metalevel knowledge to reduce constraint checking. In: *Proc.* ECAI'94.

[5] Mackworth, A.K. Consistency in networks of relations. In: Artificial Intelligence. Vol. 8. No. 1 (1977). p. 99-118.

[6] Mohr, R. and Henderson, T.C. Arc and path consistency revisited. In: Artificial Intelligence. Vol. 28 (1986). p. 225-223.

[7] Sabin, D. and Frueder E.C. Contradicting conventional wisdom in constraint satisfaction. In: *Proc.* ECAI'94.

[8] Valencia, F. and Castaño, G. Problemas de satisfacción de restricciones: Unificación y nuevos algoritmos. AV-95-03, Grupo AVISPA, Universidad Javeriana de Cali, 1995.

[9] Van Hentenryck, P. and Deville, Yves. An efficient Arc Consistency Algorithm for a Class of CSP Problems. p. 325-330. In: *Proc.* IJCAI'91.

[10] Van Hentenryck, P. Constraint Satisfaction in Logic Programing. MIT Press, 1989.

[11] Waltz, D. Generating semantic descriptions from drawing of scenes with shadows. Technical Report A1271, MIT, MA, 1972.

UN MICROMUNDO PARA LA RESOLUCIÓN DE PROBLEMAS DE DEMOSTRACIÓN EN GEOMETRÍA

Vanda Luengo
e-mail: Vanda.Luengo@imag.fr
tel : (+33) 76 51 42 45
fax : (+33) 76 51 45 55
EIAH, Laboratoire Leibniz, IMAG
B.P. 53, 38041 GRENOBLE CEDEX 9
FRANCE

RESUMEN

El objeto de esta investigación es diseñar e implementar a partir de Cabri-Géomètre (programa de construcción de figuras geométricas), los principios de un micromundo de prueba, permitiendo al estudiante la interacción con un agente racional, que él utilizará para exprimir los conocimientos que desea aplicar para resolver el problema, para construir la prueba. La función del agente racional es de asegurar las retroacciones con respecto a la coherencia propia al razonamiento del estudiante y no a una concepción preestablecida de lo que debería ser la solución. El estudiante exprimirá su conocimiento a partir de enunciados a los cuales él podrá darle diferentes estatutos (conjetura, lema, hipótesis, teorema, etc.). La interacción entre el micromundo y el estudiante esta basado en el principio de aprendizaje colaborativo.

1. INTRODUCCIÓN

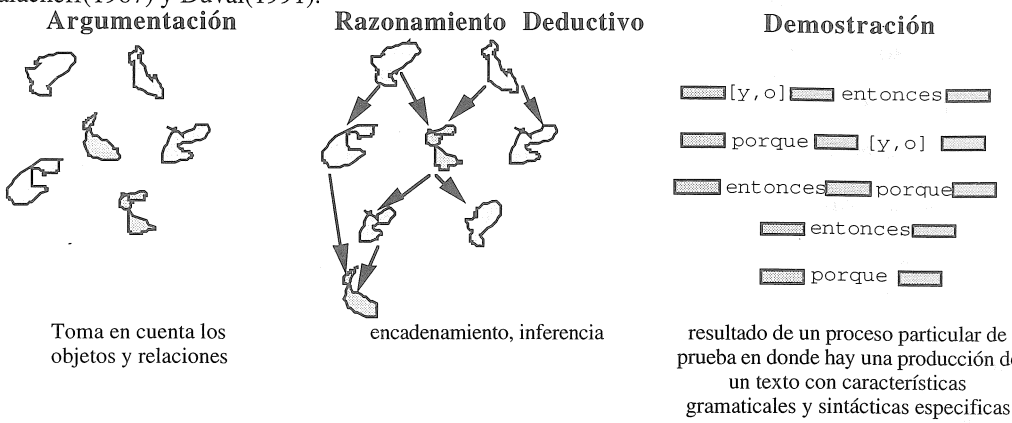
La demostración es un área de investigación bastante activa, tanto en didáctica, como en inteligencia artificial. Las investigaciones en didáctica han determinado la complejidad del problema desde el punto de vista de la enseñanza y aprendizaje de la demostración. Podemos distinguir cuatro aspectos de esta complejidad: la devolución de los estudiantes con respecto a la demostración (Brousseau, 1986), el contrato didáctico (Legrand 1990), la relación entre prueba y demostración (Balacheff 1987, Arsac 1988, Hanna 1993, Duval 1992), los aspectos cognitivos y lingüísticos de la demostración (Duval 1991). Las investigaciones en inteligencia artificial están esencialmente centradas sobre la concepción y realización de ambientes informáticos de aprendizaje humano, ya sea desde la perspectiva de desarrollar útiles de informática en relación a la modelización del conocimiento o del razonamiento, ya sea con la perspectiva de la implementación de una teoría de aprendizaje, un ejemplo representativo en esta área es los trabajos de J. R. Anderson en Carnegie-Mellon (Anderson y al. 1990), en Francia podemos citar los trabajos Py (1990) apoyándose en la teoría de reconocimiento de planes para el diagnostico del trabajo del estudiante, los trabajos de Bazin (1993) con respecto a los meta conocimientos con respecto al análisis de una figura geométrica y finalmente Ferneda (1992) que modela un Agente Racional refutable en geometría.

Nuestro interés es pluridisciplinario en el sentido que nosotros tenemos como objetivo estudiar la problemática de la demostración desde el punto de vista didáctico, analizando algunas hipótesis del aprendizaje de la prueba, fundamentalmente la resolución de problemas como útil de aprendizaje de la demostración y la dialéctica de pruebas y refutaciones; y desde el punto de vista de la inteligencia artificial, implementando algunos aspectos tales como el aprendizaje colaborativo (collaborative learning), los modelos cognitivos y principalmente la teoría de micromundo.

2. MODELAR LA PRUEBA Y LA DEMOSTRACIÓN

Desde el punto de vista cognitivo podemos dividir la construcción de la demostración en 3 fases: Argumentación, razonamiento deductivo y demostración propiamente dicha. En la primera fase, llamada argumentación heurística, se realiza un razonamiento sobre la figura, se toman en cuenta objetos y relaciones de la figura que son particulares al problema. La segunda fase,

"Razonamiento Deductivo", define el acto de relacionar cada una de las propiedades que se han inferido a partir de la figura y la organización de estas deducciones. y la tercera fase es el resultado de un proceso particular de prueba donde hay una producción de un texto con características gramaticales, semánticas y sintácticas particulares. Este modelo esta basado en los trabajos de Balacheff(1987) y Duval(1991).



Para nosotros las dos primeras fases definen el proceso de resolución de problemas para la construcción de la prueba, la tercera fase es consecuencia de un fenómeno social, ya que es una formalización preestablecida por la comunidad matemática para aceptar una prueba (Balacheff 1989). La construcción de la prueba (las dos primeras fases) es un proceso no secuencial, en el que se construyen conjeturas, estas pueden eventualmente ser refutadas (a través de un contra-ejemplo, de un enunciado contradictorio), reestructuradas o redefinidas, para finalmente llegar a la prueba. Este protocolo es definido como proceso de pruebas y refutaciones (Lakatos 1976, Balacheff 1987, Chazan 1993).

3. EL APRENDIZAJE COLABORATIVO Y LA TEORIA DE MICROMUNDO

En la ultima década el área de inteligencia artificial en educación se ha interesado en extender los campos de IA para soportar la enseñanza uno-a-uno, creando sistemas que soportan interacciones de tipo enseñanza-aprendizaje colaborativo. Más tarde y de manera independiente se ha investigado con respecto al análisis y al desarrollo de técnicas y modelos cooperativos para la resolución de problemas (Baker & Bielaczyc 1995). La idea principal del aprendizaje cooperativo es

que tanto el aprendiz como la computadora participan en la resolución de un problema. En esta interacción hay un proceso de adaptación por las dos partes para poder lograr los objetivos propios a cada participante.

Dentro de estas lineas de investigación se puede remarcar que la calidad de la colaboración depende fuertemente de cómo el verdadero aprendiz percibe su co-aprendiz (la computadora en nuestro caso). La interacción debe ser lo mas transparente posible, la autenticidad de esta colaboración requiere que el co-aprendiz no esconda estrategias didácticas.

> " ITS is in contradiction to a discovery approach: on the one hand, we tell the student that he must acquire knowledge by himself, but on the other hand, this knowledge exist somewhere in an inspectable form " (Dillenbourg,Self 1990).

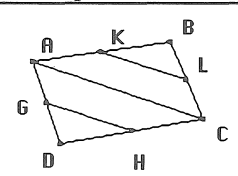
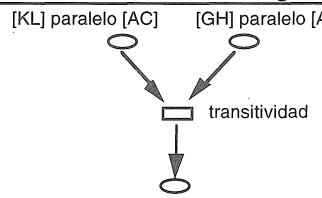
Una respuesta a la transparencia en la interacción sujeto-computadora es la teoría de micromundos. Un micromundo no impone un modelo, él no hace comparaciones con un modelo preestablecido (modelo didáctico o modelo del estudiante), el micromundo trabaja con respecto a una teoría o un subconjunto de la teoría. El micromundo es un lugar en donde el sujeto manipula un conjunto de objetos y relaciones, con la posibilidad de crear nuevos objetos y/o relaciones, estos objetos y relaciones deben ser coherentes con respecto a la teoría que hay en juego pero no con respecto a un modelo externo preestablecido. Este escenario puede ser ejemplificado con el juego Lego, cuando construimos con las piezas (objetos) de Lego, para poder ensamblar (relacionar) una pieza con una otra se deben cumplir ciertas condiciones (reglas de coherencia). Cuando ensamblamos un conjunto de piezas si estas no están construidas con ciertos principios (físicos por ejemplo), la construcción se caerá. Las intenciones didácticas pueden venir después, externamente al modelo implementado, por ejemplo Las piezas de Lego no definen nada en principio, pero uno podría pedir construir un cubo de ciertas dimensiones, con las piezas de Lego para crear una situación en donde el estudiante resuelva un problema con respecto al tema de calculo de áreas y volúmenes.

En la teoría de aprendizaje colaborativo la negociación es considerada como un medio para establecer la cooperación. La negociación implica la realización de proposiciones que pueden ser aceptadas o refutadas, se puede eventualmente trabajar con varias estrategias para buscar la

matemática relativa a un enunciado. El podrá evaluar un enunciado en función de la figura que ha sido construida en Cabri-Géomètre y eventualmente refutar el enunciado con un contra ejemplo.

4.1. Representación de los enunciados en Cabri-Euclide

El enunciado es el elemento de base de Cabri-Euclide, él exprime una propiedad de una prueba. En Cabri-Euclide el enunciado puede expresarse lingüísticamente, a través de un texto con Cabri-Euclide, o figuralmente, a través de un diseño con Cabri-Géomètre. Como lo muestra Duval (Duval 1991) el trabajar la prueba en diferentes registros ayuda a la toma de consciencia del rol de cada enunciado y su relación con los otros enunciados en la prueba, además es útil como elemento de organización en la actividad de resolución del problema, es por esto que nosotros definimos tres tipos de representación asociados a un enunciado : la figura, el grafo y el texto.

		CN: [Kl] paralelo [GH] Prueba: CJ: [KL] paralelo [AC] CJ: [GH] paralelo [AC] TH: transitividad CJ=conjectura, CN = conclusión

Estos tres contextos están correlacionados[2] en el sentido que cualquier acción sobre un contexto afecta los otros dos. La figura es construida en Cabri-Géomètre y es tiene prioridad sobre las otras representaciones, esto implica que cuando se construye un enunciado en el registro textual él debe ser coherente (el objeto debe existir, no debe provocar una contradicción) con respecto a el registro figural. El registro texto y el registro grafo son simétricos, es decir, todo lo representado en el texto esta representado el grafo, ya que el registro grafo es una imagen del texto.

4.2 El Estatuto de un Enunciado como elemento de negociación

El estatuto del enunciado es definido en función del valor "epistémico" de la proposición. El valor epistémico es el grado de certitud o de convicción que podemos asociar a una proposición(Duval [1991], pg. 255) . En un proceso de resolución de problemas podemos estar

[2]Por el momento los tres contextos pertenecen a tres aplicaciones diferentes (Cabri-Géomètre, Cabri-Graphe y Cabri-Euclide), ya existe una interconexión entre Cabri-Euclide y Cabri-Graphe pero falta la interconexión Cabri-Euclide y Cabri-Géomètre.

aceptación (Baker 1993). El isomorfismo con respecto a resolución de problemas y la construcción de la prueba, es el proceso de pruebas y refutaciones que representa la negociación para la aceptación de una conjetura con respecto a una teoría. En el caso de los micromundos son lo objetos y las relaciones que representan esos principios teóricos y la opción de crear nuevos objetos o relaciones representan la capacidad de negociación y de aprendizaje colaborativo.

Partiendo de estos principios nosotros utilizamos la teoría de micromundo, creando un ambiente para la resolución de problemas en donde el sujeto aprende durante la construcción de la prueba y el micromundo aprende a través de las construcciones hechas por el sujeto. La aceptación de las construcciones representa la capacidad de negociación entre el sujeto y el micromundo. Así nosotros definimos Cabri-Euclide con el objetivo de la gestión de conocimientos para la resolución de problemas y la construcción de la prueba.

4. DESCRIPCIÓN DEL MICROMUNDO CABRI-EUCLIDE

Cabri-Géomètre (Laborde 1989, Bellemain 1992), es un micromundo en el cual el estudiante puede construir y manipular figuras geométricas para explorar sus propiedades y descubrir fenómenos de geometría. En la enseñanza clásica hay una gran distancia entre la figura y la demostración: la figura es trazada únicamente como complemento a la retórica de la demostración, en cambio la elaboración de una demostración apoyándose en Cabri-Géomètre explota completamente la figura como un lugar de experimentación, para descubrir elementos de prueba.

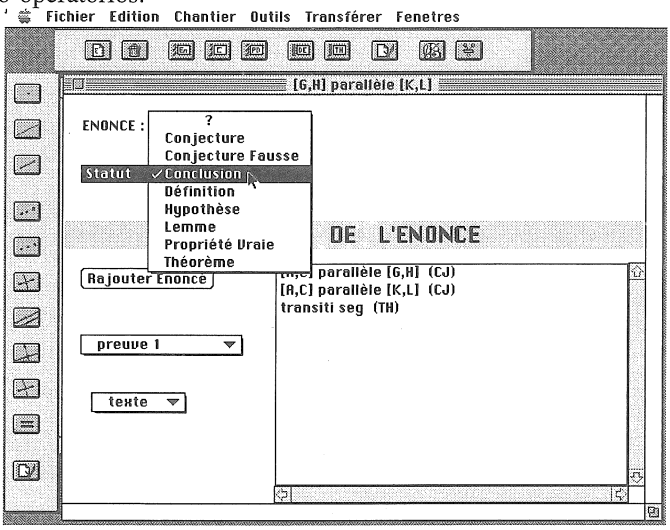
El micromundo de prueba, que nosotros llamamos Cabri-Euclide, establece un vínculo entre el análisis heurístico de la figura, realizado con Cabri-Géomètre, la formulación de conjeturas y la construcción de la prueba.

Dado un problema de geometría, el estudiante podrá construir la figura correspondiente con Cabri-Géomètre y producirá los enunciados que él desee con Cabri-Euclide[1]. Cabri-Euclide analiza la estructura del conjunto de enunciados y diagnostica si hay o no la estructura de una prueba

[1] En estos momentos Cabri-Euclide es el resultado de 2 años y medios de trabajo, existe un prototipo que esta escrito en C++.

interesados en probar un enunciado, para probarlo debemos pasar por diferentes etapas, estas etapas, o sub-problemas, pueden ser prealablemente conocidos (teoremas, hipótesis, etc.) o no (conjetura, lema), y eventualmente podemos estar interesados en probar los sub-problemas. Nosotros permitimos este proceso en Cabri-Euclide, ya que el sujeto exprime el estatuto del enunciado. Para nosotros éste es un elemento de negociación ya que a través del estatuto del enunciado Cabri-Euclide comprenderá como tratar el enunciado, y eventualmente él puede proponer el cambio del estatuto si él lo considera necesario y si la persona esta deacuerdo. El micromundo gestionará el conocimiento en función de los estatutos de los enunciados.

En Cabri-Euclide se pueden manipular dos tipos de enunciados, los enunciados "operatorios" y no-operatorios.



Los Enunciados operatorios, los cuales permiten la deducción directa de un enunciado (Duval 1991), son los siguientes : Conclusión, Hipótesis, Teorema del curso, Propiedad Verdadera y Definición. La propiedad verdadera ha sido construida con respecto a un problema especifico, podría ser eventualmente generalizada y convertirse en un teorema del curso en el caso que exista un proceso de institucionalización, lo que requiere la intervención del profesor.

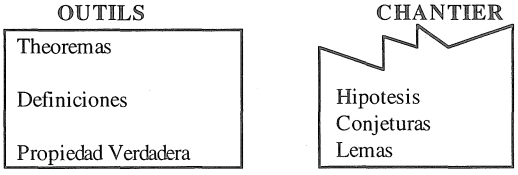
Los enunciados no-operatorios son considerados por Cabri-Euclide porque son validos en el proceso de pruebas y refutaciones (Balacheff [91]), en el sentido que el sujeto busca validar un

enunciado, usando etapas intermedias que no han sido, todavía, demostradas como verdaderas. En esta búsqueda pueden haber reformulaciones o cambios con respecto a el grado de verdad que se le asocia a un enunciado. Estos enunciados no-operatorios son la Conjetura , el Lema y la Conjetura falsa. La conjetura falsa es un enunciado que falla con respecto a una teoría, eventualmente esa falla se puede mostrar con un contra-ejemplo.

4.3 Construcción y Manipulación de los enunciados

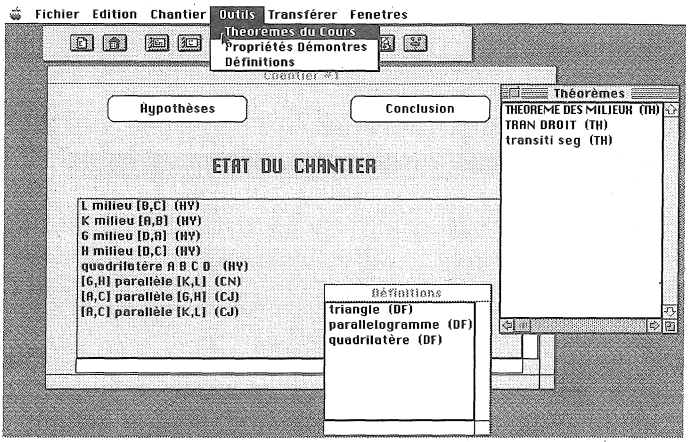
Es en la construcción y manipulación de los enunciados que se produce el proceso de negociación y la posibilidad del aprendizaje colaborativo. Para esto es importante que la creación y manipulación de los objetos sea los mas cercano a la realidad matemática que ella implica.

En la manipulación directa el ratón de la computadora es la "prolongación de la mano" que interviene para manipular físicamente objetos en la pantalla. Ella debe permitir al estudiante quedarse próximo de la acción en sus actividades con el ambiente informático operando sobre objetos formales. Para un estudiante, la necesidad de declarar sus acciones, los medios que el tiene para hacerlo y la interpretación que hace el útil informático constituyen los obstáculos y las propiedades del medio, propias de las características del modelo matemático. Es por adaptación a esos obstáculos y a la explicación de sus propiedades que él debe construir su conocimiento (Bellemain 1992, pg. 102). La interface de un micromundo debe ser lo mas transparente posible, en el sentido que los obstáculos producidos por el medio, en nuestro caso Cabri-Euclide, puedan ser interpretados y eventualmente negociados. Cabri-Euclide ofrece 2 tipos de contexto, con "capacidad de negociación" diferente: el contexto "chantier" y el contexto "outils".



El contexto "chantier", que en español quiere decir obra de construcción, permita la resolución del problema, el soporta soluciones parciales y la utilización de lemas y conjeturas. El contexto

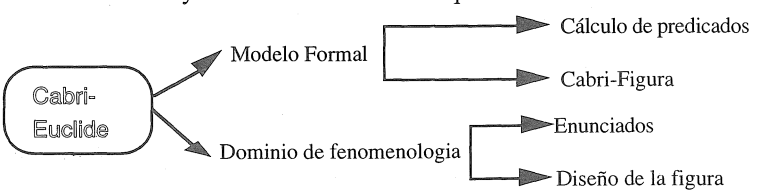
"outils", útiles, contiene los enunciados que ya están probados y que van a servir de útiles para la construcción de nuevos enunciados.



Debemos resaltar el aspecto ***macro-construcción*** del micromundo: es posible construir enunciados, y según el tipo de construcción (propiedad verdadera o definición), ellos podrán ser nuevos útiles del micromundo, es decir ellos podrán ser reinvertidos para la construcción de otro enunciado o para resolver otro problema. Por ejemplo si el micromundo conoce el concepto de paralelas y de cuadrilátero, en Cabri-Euclide podríamos definir el nuevo concepto paralelogramo. El protocolo de macro-construcción es definido por un proceso de negociación ya que, para que Cabri-Euclide acepte esa macro-construcción, el sujeto y la aplicación deben estar deacuerdo en cuanto a la coherencia y el dominio en el que es valido ese nuevo concepto.

4.4 Arquitectura del micromundo

Distinguimos dos niveles de descripción en Cabri-Euclide (Balacheff 1994 p.33), que corresponden a las dos constituyentes esenciales de su arquitectura :



1) ***El modelo formal*** describe la referencia teórica del micromundo. Para describirlo utilizamos el lenguaje formal LDL (Logical Description Language) (Desmoulins 1994 p. 53) y la figura "Cabri

Géomètre". LDL es un lenguaje de calculo de predicados que permite deducciones lógicas sobre las figuras geométricas, LDL permite manipular las expresiones y exprimir las inferencias. La figura "Cabri-Géomètre" es el enunciado de una figura que ha sido construida con "Cabri-Géomètre"

Por ejemplo, el objeto paralelogramo será representado por el siguiente predicado :

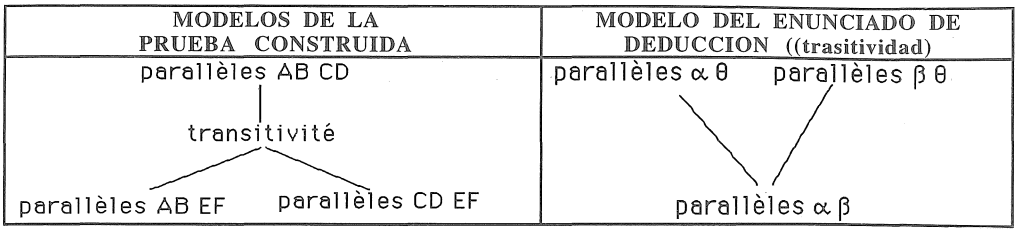
```
Paralelogramo (ABCD) =
        Paralelo(Segmento(Punto(A),Punto(B)), Segmento(Punto(C),Punto(D)))
        Paralelo(Segmento(Punto(A),Punto(D)), Segmento(Punto(B),Punto(C)))
```

2) ***El dominio de fenómenologia*** est el lugar donde se materializan las retroacciones que el micromundo produce como consecuencia de las decisiones y acciones del usuario. Las retroacciones son calculadas en el sistema formal, que exprimen configuraciones particulares de objetos o de relaciones, y son traducidos por "hechos" en la interface : un desplazamiento de puntos para producir un contra-ejemplo, un análisis sobre la coherencia de una prueba (incompleta, falsa, correcta) que se exprime en un mensaje. Estos hechos representan fenómenos cognitivos que permiten la comprensión, la explicación.

5. VERIFICACIÓN DE LA COHERENCIA DE UN ENUNCIADO

El agente racional del micromundo es lo que nosotros llamamos "verificación de la coherencia". Este módulo representa el útil de negociación entre la computadora y el sujeto. Cuando construimos una prueba y queremos que ella sea aceptada pediremos la "verificación de la coherencia" y a ese momento el agente racional nos dirá porque el esta deacuerdo o no con esa prueba. Esta verificación consiste en examinar la validez de un enunciado con respecto a su dominio. El dominio es definido en función de los otros enunciados que han sido utilizados en ese problema (contexto "chantier"), los útiles de deducción que intervienen (contexto "outils": teoremas, definiciones) y la figura que ha sido construida. La verificación es hecha en el modelo formal y se exprime en el dominio de fenómenologia.

La verificación de la coherencia con respecto a un enunciado hace la comparación entre el modelo de prueba que ha sido construido por el sujeto y el enunciado que es utilizado como útil de deducción en esa prueba (un teorema o una definición). Ambos son representados por un árbol.



Cuando se hace comparación de estos arboles se deben resolver dos tipos de problemas: la ***unificación*** de variables y la ***equivalencia*** de los dos arboles. La unificación de variables representa la problemática de que una representación es una instancia a una figura (la prueba construida) y la otra representación es generalizada (el enunciado de deducción). En nuestro ejemplo, debemos reconocer que α es AB, θ es CD, etc. La segunda problemática es dada a la equivalencia en las de propiedades geométricas, es decir [AB]= [BA]. En Cabri-Euclide estos dos problemas son resueltos a la misma vez. Primero se verifica si los dos arboles son isomorfos y si es el caso, se estable un orden geométrico con respecto a los objetos (hojas del árbol) y se trata de unificar verificando que las variables generales sean unificadas uno-a-uno con los objetos instanciados.

6. CONCLUSIÓN

El proceso de pruebas y refutaciones representa para nosotros el escenario de construcción de una prueba, con características fundamentales como la resolución de problemas y el carácter social de la prueba. Buscando el isomorfismo en la interacción computadora-sujeto, pero con componentes propias a la computadora como es la posibilidad de trabajar uno-a-uno, y de proveer elementos de organización para la resolución de problemas, en Cabri-Euclide modelamos este proceso a través del estudiante que construye una prueba, valida su construcción en interacción con el agente racional que realiza la verificación de la coherencia, con elementos de negociación como el estatuto de un enunciado y la macro-construcción.

El carácter social de la prueba nos abres las puertas a un punto de vista interesante que es la computadora como un elemento social en esta interacción. La teoría de aprendizaje colaborativo nos permite modelar el tipo de participación por parte de la computadora. Agrandar los componentes de

esta comunidad parece enriquecer el proceso y es en este sentido que nosotros buscamos trabajar en una nueva dimensión que es la telepresencia, es decir la intervención a distancia del profesor. En este caso la computadora tiene eventualmente un nuevo rol que es el de proveer al profesor toda la información concerniente al trabajo del estudiante y al problema que se debe resolver, para que el profesor pueda hacer un diagnóstico y así participar también en la resolución del problema.

BIBLIOGRAFIA

Anderson J.R., Boyle C.F., Corbett A., Lewis M. (1990) Cognitive modeling and intelligent tutoring. *Artificial Intelligence* 42, 7-49.

Almouloud S., Giorgiutti I. (1993) EIAO et didactique: le cas de DEFI, outil didactique et aide à la recherche. In : Baron M., Gras R., Nicaud J.-F. (eds) *Actes des Troisièmes Journées EIAO Cachan* (pp.15-25). Paris : Eyrolles.

Arsac G. (1990) Les recherches actuelles sur l'apprentissage de la démonstration et les phénomènes de validation en France. *Recherches en didactiques des mathématiques* 9 (3) pp. 247-280.

Balacheff N. (1987). Processus de preuve et situation de validation. Educational Studies in Mathematics 18, pp. 147-176.

Balacheff N. (1991). Treatment of Refutations : Aspects of the Complexity of a Constructivist Approach to Mathematics Learning. In : Glasersfeld E. von (ed.) *Radical Construtivism in Mathematics Education.* (pp. 89-110) Dordrecht : Kluwer Academic Publishers.

Balacheff N. (1994). Didactique et Intelligence Artificielle. Recherches en didactique des mathématiques, Volume 14/1.2. La Pensée Sauvage.

Bazin J.M.., (1993). Un modèle d'expert en résolution de problème de géométrie, Journées EIAO Cachan 1993.

Baker M. (1993). Dialogic Learning : Negotiation and Argumentation as Mediating Mechanisms. Proceedings of IA-ED '93 : World Conference on Artificial Intelligence in Education, August 1993, Edinburgh (GB).

Baker M. & Bielaczyc K. (1995). Missed opportunities for learning in collavorative problem-solving interactions. Proceedings of IA-ED '95 : World Conference on Artificial Intelligence in Education, August 1995, Washinton (USA).

Bellemain F.(1992). Conception, réalisation et expérimentation d'un logiciel d'aide à l'enseignement de la géométrie: Cabri-géomètre, thèse LSD2-IMAG, Université Joseph Fourier, Grenoble I.

Brousseau G. (1986) *Théorisation des phénomènes d'enseignement des mathématiques.* Thèse d'état. Université de Bordeaux 1.

Chazan D. (1993) High School geometry Student's Justification for their Views of Empirical Evidence and Mathematical Proof. Educational studies in Mathematics 24(4) 359-387.

Duval R. (1991). Structure du raisonnement déductif et apprentissage de la démonstration. Educational Studies in mathematics 22, pp. 233-261.

Dillembourg p. & Self J. (1990). Designing Human-Computer Collavorative Learning. Computer Supported collavorative Learning, NATO, serie F.

Desmoulins C. (1994). Etude et réalisation d'un système tuteur pour la construction de figures géométriques, thèse LGI-IMAG, Université Joseph Fourier, Grenoble1.

Ferneda E. (1992) *Conception d'un agent rationnel et examen de son raisonnement en géométrie.* Thèse. Université de Montpellier II.

Hanna G., Jahnke (1993) Proof and Application. Educational studies in Mathematics 24(4) 421-438.

Laborde J. & Triling L. (1989) Conception et réalisation d'un système intelligent d'apprentissage de la géométrie, Presentation de projet LSD-IMAG, Grenoble.

Lakatos, I. (1976). Proof and Refutations. Cambridge: Cambridge University Press.

Legrand M. (1990) Rationalité et démonstration mathématiques, le rapport de la classe à une communauté scientifique. *Recherches en didactique des mathématiques* 9 (3) 365-406.

Py D., Nicolas P. (1990) MENTONIEZH: A geometry ITS for figure drawing and proof setting. Artificial Intelligence in Education. 1(3) 41-55.

Costos de la Calidad y del Mejoramiento del Proceso de Software

Mauricio Concha F. y Marcello Visconti Z.

Departamento de Informática

Universidad Técnica Federico Santa María

Valparaíso, Chile

e-mail: {mconcha,visconti}@inf.utfsm.cl

Resumen

Cuando en una organización de software se fijan objetivos de calidad, se debe definir un plan preciso que conduzca a la obtención de tales objetivos. El plan debe contemplar el mejoramiento de los procesos de software como forma fundamental de obtener calidad. Además, los efectos del plan deben ser cuantificados a través de conceptos específicos que componen el costo de la calidad. Al utilizar esta orientación, se puede señalar que el plan de mejoramiento debe ser costo-efectivo, con el objeto de lograr calidad y productividad para la organización. Este trabajo determina los conceptos claves para lograr calidad costo-efectiva, y de qué forma debe ser utilizado apropiadamente algún método de evaluación, como el modelo CMM o las normas ISO 9000, para desarrollar un plan de mejoramiento de procesos.

1.- Introducción.

El objetivo de la Ingeniería de Software es desarrollar un producto de software de alta calidad, a partir de recursos limitados, mediante un proceso en el cual se utilizan diversas metodologías, herramientas y estándares.

La calidad del producto de software obtenido en un proyecto no es un resultado aleatorio ni dependiente de eventualidades ni esfuerzos individuales, sino que la calidad obtenida estará determinada por la forma cómo se lleva a cabo el proceso de desarrollo. Este enfoque para lograr calidad pone el énfasis en el mejoramiento y perfeccionamiento contínuo del proceso de software.

La calidad de software está determinada por la calidad del proceso utilizado para desarrollar y mantener el software. Este concepto no es más que una extensión del principio que ha sido aplicado exitosamente en la industria de manufactura y de servicios en los últimos años (5). Esto indica que para la obtención de calidad en proyectos de software la atención debe ser puesta esencialmente en el proceso más que en el producto (9).

La productividad debe considerarse como un factor económico relevante cuando se define un plan de mejoramiento de procesos. Un plan bien definido y ejecutado de mejoramiento de procesos debe, por sí solo, asegurar calidad y aumentar productividad, basándose en la idea que los costos de producción son reducidos significativamente cuando se desarrolla software con un bajo índice de errores (3).

En este trabajo se analizan los factores que relacionan el aseguramiento de la calidad con el aumento de la productividad, y como éstos se reflejan en un plan de mejoramiento del proceso de software basado en el Capability Maturity Model (CMM) o en la certificación de calidad ISO-9000.

2.- Factores Económicos de la Calidad.

Cuando se fijan objetivos de calidad, se hace necesario que las organizaciones midan los resultados de sus esfuerzos destinados al mejoramiento de los procesos. Un enfoque adecuado para cuantificar los efectos de las iniciativas de mejoramiento es el enfoque basado en la idea de "costo de la calidad" de Phil Crosby (5). Extendiendo este enfoque, se pueden definir los costos de desarrollo como los costos de realización más los costos de la calidad. A partir de esto, se hace diferencia entre el costo de realizar algo correctamente la primera vez y el costo de rehacer el trabajo

Costos del Desarrollo = Costos de Realización + Costos de Calidad

En donde,

1.- Costos de Realización: Asociados con hacer las cosas correctas la primera vez. (Especificación de requerimientos, diseño, codificación).

2.- Costos de la Calidad:

2.1.- Costos de Prevención: Asociados a las actividades que intentan prevenir fallas antes que puedan llegar al producto.

2.2.- Costos de Evaluación: Asociados con revisiones e inspecciones de los productos generados durante el proceso de software.

2.3.- Costos de Corrección o de Rehacer el Trabajo (*rework*) debido a la ocurrencia de defectos.

Entre los costos de prevención se pueden destacar los costos de desarrollo de prototipos. Los prototipos son usados para lograr un claro entendimiento de requerimientos, funcionalidad o estructura del software a construir. El principal objetivo del desarrollo de prototipos es evitar incurrir

en errores, que en fases posteriores provocarían grandes costos de corrección (4).

Los costos de evaluación son asociados a esfuerzos destinados a revisiones e inspecciones formales.

Los costos que están asociados a la corrección de fallas y errores se clasifican como costos de *rework* o de rehacer trabajo, y usualmente están ocultos como trabajo real en muchos proyectos. Estos costos deben ser considerados como claves cuando se definen programas de mejoramiento de procesos.

Los costos de calidad son una herramienta de administración y proporcionan una visión de cómo la organización está funcionando. Si estos costos son altos, la organización está probablemente funcionando en forma ineficiente con ciclos de vida extensos, baja productividad y costos más altos de lo necesario. Por otro lado, bajos costos de calidad indican un proceso de desarrollo optimizado que se logra haciendo las cosas correctamente.

Es posible relacionar los costos de desarrollo con mediciones de tamaño de un producto, en puntos de función por ejemplo, con el fin de obtener datos de productividad para un ambiente de desarrollo. Según estos conceptos, obtenemos la siguiente relación:

Productividad: Tamaño del Producto / (Costos de Realización + Costos Calidad)

La siguiente relación define la variación de productividad:

$$\Delta P = \frac{T}{R + \Delta(r + p + e)}$$

En donde, P = Productividad.
R = Costos Realización.
r = Costos *rework*.

p = Costos prevención.
e = Costos evaluación.
T = Tamaño software

La variación de la productividad para un software de tamaño T estará dada principalmente por la variación del costo de la calidad (r+p+e).

Se considera el costo de realización (R) como constante o con una variación despreciable, debido a que a este costo no asocia los recursos asignados a detección y corrección de errores, principal razón de la variación de los costos de desarrollo de un proyecto.

A partir de esto, con el objeto de aumentar calidad y aumentar productividad se debe dar:

$$\Delta (r + p + e) <= 0$$

Con las siguientes condiciones:

$\Delta r < 0$, disminución costos de *rework*.

$\Delta p >= 0$, aumento costos de prevención.

$\Delta e >= 0$, aumento costos evaluación.

Por lo tanto, para obtener calidad costo-efectiva se debe cumplir estrictamente:

$$| \Delta r | >= | \Delta p | + | \Delta e |$$

Un sistema de calidad costo-efectivo conduce a incrementar la productividad y a una permanente reducción de costos, debido a que posibilita a la administración del proyecto reducir los costos de *rework* o de corrección de errores al colocar el énfasis en la prevención y en la revisión.

Los costos iniciales de establecer un sistema de calidad son compensados por los posteriores resultados logrados dentro de uno o dos años. Por ejemplo, *Raytheon Equipment Division* fue capaz de reducir los costos por fallas en un periodo de tres años desde 45% a 15%, y el costo de la calidad

se redujo al 20% de los costos del proyecto, todo esto logrado al mejorar sus procesos de software (5).

Es interesante señalar que en 1987, Barry Boehm estimó que eliminando el rehacer trabajo se podría reducir los costos del software entre 30% a 50% (8).

Reduciendo *Rework*.

El *rework*, que son las actividades asociadas con rehacer trabajo, posee varias componentes. De este modo es de valor analizar cada factor de esfuerzo o costo que contribuye principalmente a la reducción global de *rework* y tratar de determinar cuáles cambios en los procesos podrían lograr estas reducciones.

Además, importantes ahorros en costos de *rework* se pueden lograr a partir de aumentos leves en el costo de otros procesos. Por ejemplo, en el caso de la compañía *Raytheon* en EE.UU. (5), en donde se desarrolló un completo plan de mejoramiento de su proceso de software, los costos de diseño y codificación se elevaron levemente debido a que las inspecciones formales reemplazaron a las revisiones informales. Esta acción de cambio, llevó a ahorros en *rework* debido a que permitió descubrir problemas en el código antes de la fase de integración de software, eliminando de esta manera el *retesting*.

Con el objeto de reducir el *rework* se debe conocer cuáles son las actividades que contribuyen más fuertemente a los costos de éste, es decir, qué tareas dentro del desarrollo se deben volver a hacer debido a defectos.

Detección y Corrección Temprana de Defectos.

El número de defectos puede ser reducido sustancialmente al adoptar una estrategia para prevención de defectos, pero éstos no serán eliminados completamente. Por esta razón, se debe

detectar y corregir los defectos tan pronto como son introducidos en el desarrollo. Mientras más tiempo transcurre y un defecto no es detectado, más caro será corregirlo. Algunos datos disponibles señalan que un defecto de diseño puede llegar a ser 15 veces más caro corregirlo en la fase de *testing* que en la fase de diseño (7).

Inspecciones Formales.

A través de la revisión de los factores presentados, se puede señalar la importancia de las inspecciones para lograr reducir los costos de *rework*, basándose en que la detección y corrección de defectos debe hacerse en la misma fase en que suceden. Como ejemplo, en Hewlett-Packard (HP) se estimó que aproximadamente un tercio de todos los costos de software son provenientes de *rework*, y las inspecciones pueden ahorrar el 60% de tales costos (4).

***Testing*.**

El principal factor que afecta al *testing* es el número de defectos en el producto al comenzar esta fase. Ante presiones por cumplimiento de plazos y conociendo que se ha desarrollado un producto aún con defectos, se puede optar por prolongar los tiempos de entrega o liberar un producto de baja calidad. La solución económica a este problema es lograr remover la mayor cantidad de defectos antes de comenzar el *testing* (6). Es deseable encontrar y corregir los defectos antes que se comience el *testing*, pero eliminar todos los defectos es imposible, por lo que esta fase de *testing* requiere una importante planificación con el objeto de lograr un software de calidad.

3.- Plan de mejoramiento de procesos.

Para lograr entender cuáles mejoras son más urgentes para una organización, debe comprenderse inicialmente las diferencias entre una organización inmadura y una organización madura.

Una organización inmadura no posee una base objetiva para evaluar la calidad del producto. Existe un pequeño entendimiento de cómo las fases del proceso de software afectan a la calidad, y por lo tanto la calidad del producto es difícil de predecir. Más aún, las actividades que pretenden mejorar la calidad, como revisiones o *testing*, son a menudo eliminadas cuando los proyectos están generalmente fuera de plazos. En este tipo de organizaciones se desarrolla con altos riesgos, baja productividad, altos costos y calidad incierta.

Una organización madura posee la habilidad para administrar el desarrollo y mantención de software. Los administradores se preocupan de la calidad del producto y de los procesos. Existe una base objetiva para evaluar la calidad y analizar problemas. Plazos y presupuestos se basan en estimaciones correctamente realizadas sobre datos históricos. En este tipo de organizaciones se desarrolla con bajo riesgo, alta productividad, dentro de presupuestos y con aseguramiento de calidad (2,10).

El camino en el mejoramiento de procesos debe estar dado desde la inmadurez hacia un mayor grado de madurez, en un proceso de mejoramiento contínuo. Al revisar la relación entre la productividad y los costos de la calidad, se puede generalizar y determinar tres fases en el mejoramiento de los procesos de software por los cuales debería evolucionar una organización:

FASE 1	Costos de calidad altos dados por un gran costo en rehacer trabajo y en costos de evaluación dados por *testing*. Se obtiene baja productividad y calidad incierta.
FASE 2	Se mantiene el costo de calidad desde el estado anterior, pero la relacion entre los costos de prevención, evaluación y de rehacer el trabajo cambian. Se disminuye el costo de rehacer el trabajo y se aumentan los costos de prevención y de evaluación, con lo que se logra un producto de mayor calidad y se eliminan riesgos; la productividad no disminuye pero aumenta la calidad del proceso de software.

FASE 3	Disminución de los costos de calidad, dado por el ahorro de recursos en rehacer el trabajo. Se mantiene los costos de prevención y evaluación con lo que se asegura calidad en el producto final. La productividad aumenta y se asegura la calidad..

A partir de esto, una organización inmadura que intenta asegurar la calidad corre el riesgo de que aumente los costos del proyecto, y por consecuencia disminuya la productividad. El éxito de una organización se logra cuando la obtención de calidad va junto al aumento de la productividad, y esto se observa en donde el rehacer trabajo o *rework* ha sido reducido casi completamente. Los efectos del mejoramiento progresivo de los procesos de software versus los costos de desarrollo se ven graficados en la Figura 1.

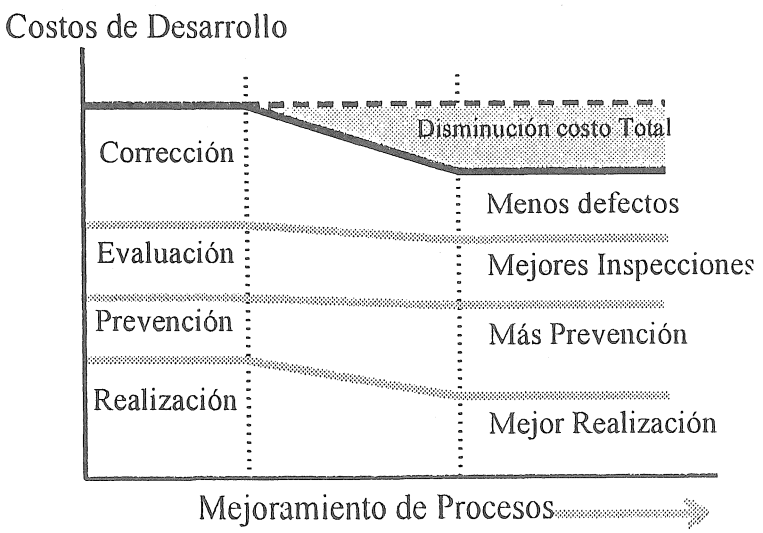


Figura 1.- Relación entre los costos de desarrollo durante el mejoramiento de procesos.

Evaluación de Procesos de Software

Existen esencialmente dos razones por las cuales las organizaciones que desarrollan proyectos de software efectúan alguna evaluación de sus procesos de desarrollo, basándose en estándares que

sean objetivos y ampliamente aceptados :

1) Obtener ventajas competitivas

2) Definir objetivos estándares para elaborar un plan de mejoramiento de procesos.

Sin embargo, el principal objetivo de la evaluación de los procesos de software en una organización debe apuntar hacia obtener un sistema de calidad costo-efectivo. Este objetivo debe ser la base de un plan de mejoramiento de procesos que se oriente en los factores económicos de la calidad (costo de la calidad, productividad).

Al detallar un plan en actividades se debe tomar como orientación algún estándar ampliamente reconocido, como por ejemplo ISO 9000 o el *Capability Maturity Model* (CMM), el cual preste una evaluación objetiva y ampliamente aceptada a los logros obtenidos en el desarrollo del plan.

Cuando se ha definido cómo lograr un sistema de calidad costo-efectivo, entonces se puede incorporar actividades específicas con el fin de lograr la certificación deseada. La Figura 2 ilustra esta relación.

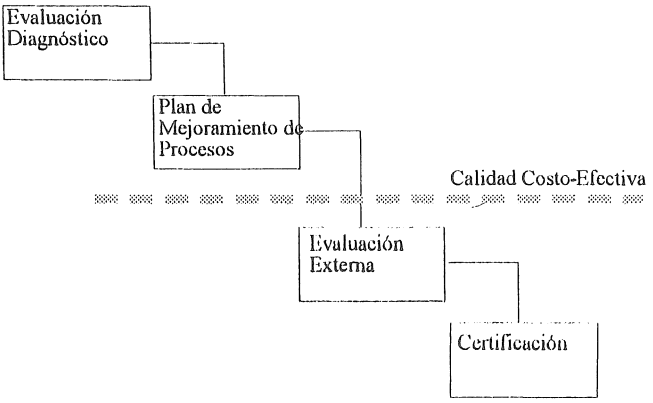


Figura 2.- Fases para lograr Calidad Costo-Efectiva y luego Certificación.

Normas ISO-9000 y Modelo CMM.

El *Capability Maturity Model* desarrollado por el *Software Engineering Institute* (SEI), y la serie de estándares ISO 9000 desarrollado por la *International Organization for Standarization* (ISO), son estándares de evaluación y certificación que poseen un interés común en la calidad y en la administración de procesos de software (10).

El estándar específico en la serie ISO 9000 que interesa cumplir a las organizaciones que desarrollan software es ISO 9001, que define normas para las etapas de diseño, desarrollo, producción, instalación y servicio. Como ayuda ISO proporciona una guía específica para la aplicación de ISO 9001 al desarrollo de software denominada ISO 9000-3.

El CMM define un marco de referencia para la madurez de los procesos de software y define 5 niveles, en los cuales cada nivel proporciona una base para lograr los niveles superiores.
Los conceptos que se manejan en ambos estándares son similares, pero la diferencia entre ellos radica en su filosofía fundamental. ISO 9001 identifica los requerimientos mínimos para un sistema de calidad, mientras CMM se basa en la necesidad de un contínuo mejoramiento de procesos a través de la evaluación de la madurez de los procesos (6).

Por dicha razón, es posible señalar que ISO 9001 conduce en forma implícita a un mejoramiento de los procesos a través de contínuas correcciones con el fin de buscar la certificación, mientras CMM es explícito en su objetivo del contínuo mejoramiento de procesos.

Lo esencial para una organización es enfocar sus esfuerzos de mejoramiento en lograr un verdadero sistema de calidad costo-efectivo, en vez de solo esforzarse en lograr una marca, ya sea ésta un nivel de madurez o una certificación.

Existen esfuerzos por parte de ISO para combinar características de CMM e ISO 9000 y otros

métodos como *Bootstrap* de ESPRIT, con el fin de desarrollar un conjunto de estándares denominado SPICE (*Software Process Improvement and Capability dEtermination*). El estándar SPICE se orienta principalmente en el proceso de software, pero se extenderá también a temas como recursos humanos, tecnología, prácticas de gestión, apoyo al cliente. SPICE es desarrollado bajo ISO/IEC/JTC1/SC7, cuerpo responsable de estándares en Ingeniería de Software , en contraste con ISO 9000 que fue desarrollado por ISO TC176, el cual es responsable de estándares para sistemas de administración de calidad en general (10).

Definición de un Plan.

Cuando se desarrolla un plan de calidad orientado por un estándar, si éste es implementado en forma errónea puede ser más negativo que positivo para la organización.

Es posible caer en prácticas que solo conduzcan a llenarse de papeles y de definiciones extensas de tareas. Para evitar esto, la atención debe ser puesta en el centro del problema, es decir, aplicar los conceptos del estándar como una guía para desarrollar software sin defectos desde las primeras fases del proceso de software.

4.- Conclusiones y Proyecciones Futuras.

Ante la exigencia de un mercado cada día más competitivo, se hace necesario producir software de alta calidad. Con este objetivo una organización debe desarrollar un plan que conduzca a la obtención de software de calidad, cuya orientación debe estar puesta en el mejoramiento contínuo de los procesos de desarrollo de software y debe basarse en estándares de evaluación que sean objetivos y ampliamente aceptados.

Un plan de mejoramiento basado en un estándar ya sea ISO 9000, CMM u otro, debe ser costo efectivo en primer lugar, para así lograr convertir a la organización en una organización madura y

lograr de este modo los niveles de calidad y productividad esperados.

La obtención de metas de calidad significará un costo, al cual se ha denominado costo de la calidad, que define un enfoque adecuado para cuantificar los efectos de las iniciativas de mejoramiento considerando el factor económico de la calidad.

¿ Cómo lograr calidad costo-efectiva ?. La respuesta está dada en la descomposición de los costos de la calidad y en la determinación de los factores que pueden influir significativamente en tales costos y que pueden ser evitados. El factor preponderante corresponde al *rework* o rehacer trabajo al corregir errores detectados.

El plan organizacional debe orientarse a lograr calidad costo-efectiva. Para conseguir ese objetivo debe reducir principalmente los costos de *rework*, lo que implica incorporar revisiones e inspecciones formales a las fases del desarrollo, evitando de este modo que los defectos permanezcan en el proceso y deban ser corregidos en forma posterior, con el conocido aumento de los costos por detección y corrección tardía de defectos. Estos conceptos deben estar presentes en la definición del plan de mejoramiento.

Es importante considerar que el logro de una certificación no es un fin en sí mismo, sino que debe ser utilizado como una orientación para lograr el cumplimiento del principal objetivo: el mejoramiento de procesos, basándose en un plan que considere conceptos de costo de calidad y de productividad. Esto con el fin de obtener un proceso de software que sea realmente costo-efectivo.

Se consideran a futuro estudios referentes al desarrollo de un modelo de plan de mejoramiento de procesos, basado en estándares de evaluación y certificación, que sea aplicado a organizaciones inmaduras, y que a través de un conjunto de métricas pueda ser evaluado en las distintas fases de la evolución de los procesos del software. Además, es necesario continuar el estudio en lo referente a

la priorización dentro de la Ingeniería de Software de los conceptos de "costo de calidad" , "calidad costo-efectiva" y "evaluación y certificación de calidad".

5. - Referencias.

1.- Norma Chilena Nch-ISO 9000-3 1994

2.- M. Paulk y B.Curtis, "Capability Maturity Model, version 1.1" IEEE Software, pág. 18-26, Julio 1993.

3.- M. Rosch, "Getting the Best Out of ISO 9000" Objects in Europe, pág. 13-15, Verano 1994.

4.- W. Humphrey, T. Snyder, "Software Process Improvement at Hughes Aircraft" IEEE Software, pág. 11-23, Julio 1991.

5.- R. Dion, "Process Improvemenet and the Corporate Balance Sheet" IEEE Software, pág. 28-35, Julio 1993.

6.- W. Humphrey, "A discipline for software Engineering" Addison Wesley Publishing Company, 1995.

7.- E. Bryan, "CP-6: Quality and Productivity measures in the 15 year life cycle of an Operating System", Software Quality Journal, pág. 129-144, 1993.

8.- B. Boehm, " Understanding and Controlling Software Costs" IEEE Transactions on Software Engineering", pág. 1462-1475, Octubre 1987.

9.- J. Gray, " Quality Costs: A Report Card on Business" Quality Progress, Abril 1995.

10.- CMU/SEI, "The Capability Maturity Model, Guidelines for improving the software process". Addison Wesley Publishing Company, Inc. 1995.

DesProc 1.0.
Una herramienta para descomposición formal funcional de procesos

Tatiana Arias Manrique
Javier Enrique Mora Díaz
Ricardo Llamosa-Villalba
Grupo de Investigación en Ingeniería del Software
Escuela de Ingeniería de Sistemas e Informática
Universidad Industrial de Santander
E-mail: rllamosa@uiscol.uis.edu.co
Apartado Aéreo 678, Ciudad Universitaria UIS
Bucaramanga, Colombia

RESÚMEN

"Descomponer Procesos", es comúnmente difícil y complejo [2][3][5][6]; la cantidad de variables, el conocimiento profundo del discurso del problema en evaluación y la subjetividad de las especificaciones, respecto a la constitución de un proceso, desencadena innumerables alternativas de modelado. El estudio del álgebra de Adler, facilita y formaliza, la labor de descomposición de procesos a través del flujos de datos. Este instrumento, con reglas claras y concisas, obtiene una misma descomposición, eliminando la influencia del criterio personal[1]. Este trabajo facilita la investigación de la descomposición de procesos de diversa índole, haciendo énfasis en procesos de ingeniería del software. Este hecho nos llevó a fabricar una herramienta que apoya la descomposición de procesos. Esta "máquina" con resultados gráficos y algebraicos, permite verificar la eficiencia y eficacia[2] de los criterios expuestos por Adler. Las debilidades y fortalezas descubiertas en nuestro trabajo de investigación se mencionan en este documento. Particularmente, la obtención de sentencias minimales[3].

1. FUNDAMENTACION

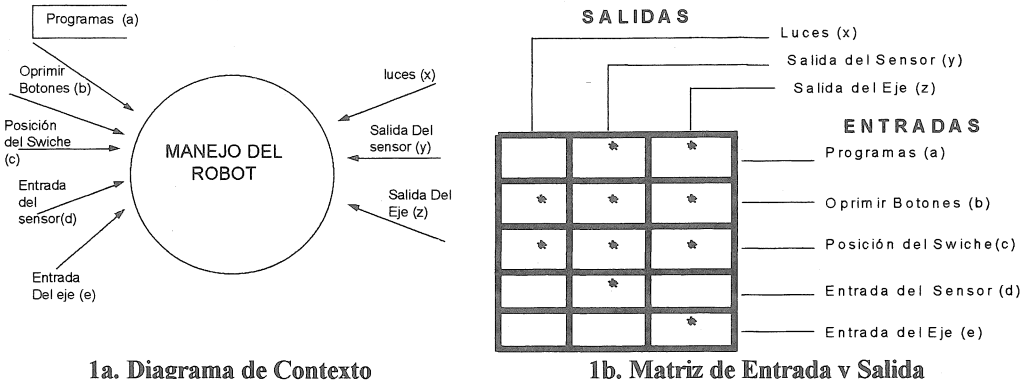
La descripción inicial de un proceso se da en un diagrama funcional denominado: "diagrama de contexto", que incluye variables de entrada, la función de contexto (caja negra) y las variables de salida. Realizar la descomposición, implica: 1) encontrar las relaciones existentes entre las variables de entrada

[1] Significa, desaparecer la posibilidad de hacer pruebas, para observar que descomposición resulta mas satisfactoria y plantear, así, el problema de otra forma, en vez de hacer diferentes descomposiciones para un mismo problema.

[2] Se pretende validar las reglas de Adler, en el proceso de interpretación, para todos los modelos que se propongan, es decir, si con ellas, en verdad, se pueden determinar, que elementos, son redundantes al final de dicho proceso. El control parte del estudio de casos con la asesoría de personas conocedoras de dichos casos.

[3] Una sentencia es minimal cuando una variable de salida se produce por uno y solo un proceso. Las sentencias no minimales no son contempladas en el álgebra de Adler.

y las variables de salida en una "matriz de entrada y salida": Las relaciones indican: que entradas son necesarias para la obtención de cada una de las salidas definidas[4]. 2) Definidas las entradas, las salidas y las relaciones, se aplican una serie de operadores algebraicos, propuestos por Mike Adler[5] [1]; Sustitución exacta, Sustitución de subconjunto, Sustitución débil, Absorción, Colección y Conexión, buscando obtener una sentencia minimal[6]. De no obtenerse una sentencia minimal, se lleva a cabo el proceso de eliminación de variables[7] [8]. 3) Obtenida la sentencia minimal se verifica la calidad de la descomposición; denominada "Interpretación". La interpretación consiste en aplicar un conjunto de reglas para obtener las sentencias iniciales dadas en la matriz de relaciones. Hecho que puede lograrse o no. La interpretación se hace sentencial o matricialmente. Un ejemplo sobre la funcionalidad de un robot, nos permite conceptualizar el modelo: La figura No. 1a, presenta el "Gráfico de Contexto" con las variables de entrada: *a, b, c, d, e* y las variables de salida: *x, y, z*. La figura No. 1b, describe las relaciones entre las variables de entrada y las variables de salida.



1a. Diagrama de Contexto **1b. Matriz de Entrada y Salida**

Figura No. 1 Contexto Funcional de un proceso

Cada una de las relaciones definidas se denota de la siguiente manera: (a->y,z), (b->x,y,z), (c -> x, y, z), (d ->y), (e -> z); sentencias iniciales que corresponden a la representación algebraica de la matriz. Aplicar los operadores tiene el siguiente orden: 1) Substitución exacta; 2) Colección; 3) Substitución de

[4] Relación tipo "es utilizada para producir".

[5] Mike Adler, es consultor y diseñador de Software de procesos. Trabaja en el desarrollo de sistemas operativos y es miembro de la IEEE: Computer Society y de "The Association for Computing Machinery".

[6] Una sentencia no minimal refleja una descomposición, en la cual, se obtienen subprocesos diferentes con salidas idénticas, lo que indica que dicha descomposición, no es válida.

[7] Mike Adler no considera la posibilidad de llegar a sentencias no minimales al final del proceso de descomposición. Ante tal la ausencia de orientación en este sentido, los autores de este proyecto definieron pautas para la obtención de sentencias minimales cuando no obtienen directamente.

[8] Siempre y cuando, se esté de acuerdo en que las relaciones no afecten la minimalidad de la sentencia.

Subconjunto; 4) Conexión; 5) Absorción; 6) Substitución Débil. Los operadores se aplican a cada transformada o par de transformadas, en la lista de transformadas, para producir la sentencia minimal. En cada lista, de entrada y salida, de las nuevas sentencias, que vayan apareciendo, se les re-aplica los operadores. Después de cada minimización, la transformada o par de transformadas, resultantes, se reemplazan en la lista de transformadas por la sentencia minimizada, y los operadores son re-aplicados a la lista de transformadas para continuar con el proceso de minimización. Aplicando el proceso, tenemos:

Operación	Entrada	Sustitución	Resultado
1)Substitución Exacta	(a->y,z),(b->x,y,z),(c-> x, y, z), (d->y), (e->z)	(b->x,y,z)<| {x,y,z <- (c-> x,y,z) }	(a->y,z),(b->(c->x,y,z)),(d->y), (e->z)
2)Colección	Resultado anterior	(b->(c->x,y,z)) < | {b}	(a->y,z), (b,c->x,y,z), (d-> y), (e->z)
3)Subconjunto	Resultado anterior	(b,c->x,y,z)<|{y,z<-(a-> y,z)}	(b,c->x, (a->y,z)),(d->y), (e->z)
4)Conexión	Resultado anterior	(b,c->x,(a->y,z))<|{(d->y) }	(b,c->x, (a->y,z), (d->y)), (e->z)
5)Subconjunto	Resultado anterior	(a->y,z) <| { y<-(d->y) }	(b,c->x, (a->z, (d->y))), (e->z)
6)Conexión	Resultado anterior	(b,c->x, (a->z, (d->y))) <| {(e->z)}	(b,c->x,(e->z),(a->z,(d-> y)))
7)Subconjunto	Resultado anterior	(a->z,(d->y))<|{z<-(e->z) }	(b,c->x, (a-> (e->z), (d-> y)))

Terminada la descomposición, la sentencia minimal se Interpreta;

a) (b, c ->x, (a-> (e ->z), (d->y))). b) (a-> (e ->z) , (d->y)). c) (e ->z). d) (d->y)

Ahora, determinamos, los nodos procesos:

Nodo Proceso	Nodo Proceso	Nodo Proceso	Nodo Proceso
a) (b,c->x)	b) (a->null)	c) (e->z)	d) (d->y)

Los nodos proceso enlazados resultantes son:

Enlace	Enlace	Enlace	Enlace
a) (b,c->x)=>(a-> null)	b)((a->null)=>(e->z),(d->y))	c) (e->z)	d) (d->y)

Esto especifica los siguientes flujos locales:

Desde el nodo Proceso		Al Nodo Proceso	
1	(b,c->x)	2	(a-> null)
2	(a-> null)	3	(e->z)
2	(a-> null)	4	(d->y)

El gráfico de interpretación resultante es el DFD que se muestra en la Figura 2. El analista nombra los procesos y los flujos de datos locales para producir el DFD final de la descomposición.

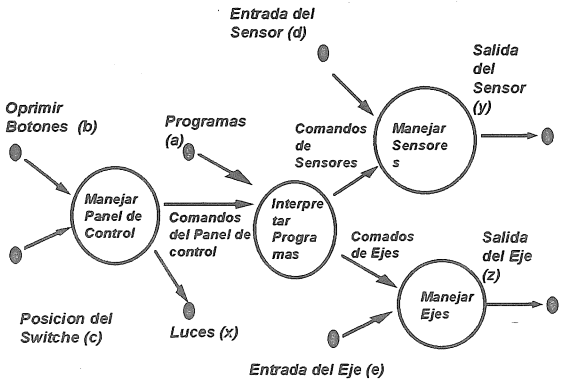


Figura No. 2 Grafo de Interpretación

Analizamos, ahora, la calidad de la descomposición, para verificar si aparecen nuevas relaciones; Esta calidad puede ser trivial, optima o factible[9]. Una descomposición trivial es idéntica a la sentencia inicial e implica la no aplicación de operadores. Una descomposición óptima tiene una única solución y la descomposición factible tiene múltiples soluciones; resulta cuando se debe usar alguna sustitución débil no conservativa o el operador de conexión (genera parejas que no necesariamente existían).

2. MODELO DE INGENIERIA

2.1. Recursos u Objetos.

En el diseño se ha considerado la necesidad de definir objetos que permitan el manejo de la información e intervengan en la descomposición de un proceso, tal como se ha explicado en la fundamentación. El propósito de cada uno de estos objetos se describe a continuación:

Objeto	Operaciones
Gráfica de Contexto	Definir, eliminar, añadir, ordenar, nombrar y actualizar las variables que intervienen en un proceso.
Matriz de Relaciones	Permitir visualizar, crear y modificar las relaciones que se consideren necesarias en la matriz de relaciones.
Matriz de Interpretación	Visualizar e identificar relaciones actuales y nuevas entre las variables de Entrada y Salida producto de la interpretación.
Sentencias iniciales	Visualizar sentencias algebraicas y las relaciones entre las variables de entrada y salida definidas en la matriz inicial.

[9] Una calidad es trivial si la relación es idéntica a la sentencia inicial o es un término. Una calidad es óptima si la relación es equivalente a la inicial y es no trivial. Una calidad es factible si la relación es no equivalente a la sentencia inicial y es no trivial.

Objeto	Operaciones
Flujo de Datos	Mostrar la descomposición obtenida en forma de diagrama de flujo de datos de tal manera que se puedan apreciar los flujos , las variables de entrada y de salida de cada uno de los nodos y el orden de precedencia de los mismos.
Interpretación	Visualizar la interpretación algebraica comparando las sentencias iniciales con las finales para establecer las relaciones adicionales generadas con la descomposición.
Variables	Identificar y Definir estructuras para representar diferentes conjuntos de variables de entrada y salida.
Relaciones	Identificar y mostrar la jerarquía de los procesos respecto a cada una de las relaciones definidas entre las variables de entrada y de salida.

2.2. Macroprocesos.

Describimos, a continuación, cada Macroproceso (figura No. 3) en los que se encuentran cada uno de los procesos diseñados para DesProc 1.0.

Administración de Archivos: Operaciones para crear, recuperar, guardar, renombrar o ubicar archivos y operaciones para la selección de equipos de impresión e impresión del contenido de la ventana activa.

Edición: Operaciones de edición y copia de cualquier presentación, eliminación y selección o inhibición de variables de presentación del "gráfico de contexto".

Visualizar presentaciones: Operaciones para visualizar gráficos de contexto, matriz de relaciones, sentencias iniciales, descomposición algebraica, DFD e interpretación de sentencias y matrices.

Creación de Variables: Operaciones para insertar variables de entrada y variables de salida en el gráfico de contexto.

Manejo de Ventanas: Operaciones para la organización y gerencia des ventanas.

3. PROTOTIPO

A partir del diseño concebido y descrito en 2, se desarrolló la herramienta en Borland Pascal (ver.7.0) para Windows. El prototipo concretó los procesos de gráfico de contexto, matriz, ventana de sentencias minimales, descomposición en sentencias, procesos y DFD's, interpretación sentencial, interpretación y representación matriz y ayudas.

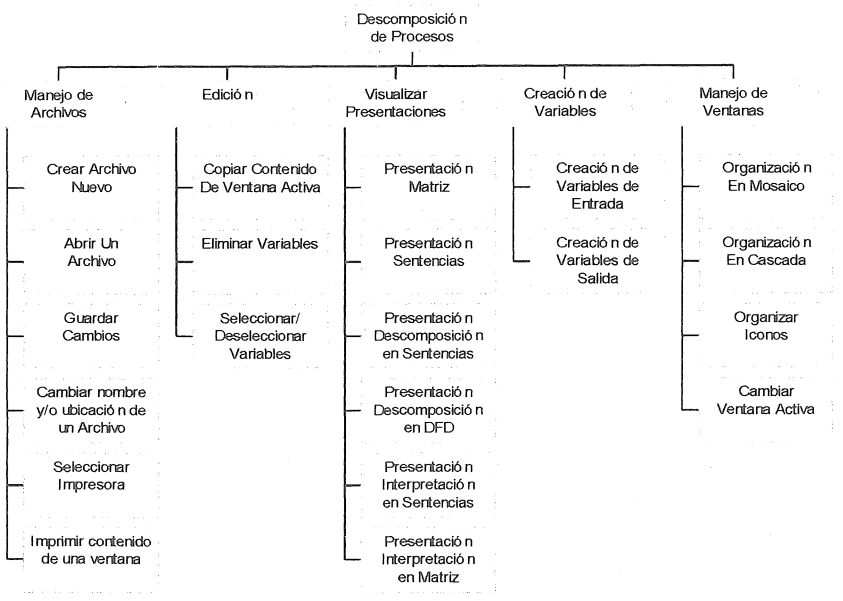


Figura No. 3 : Macro Procesos de DesProc 1.0

3.1. Gráfico de Contexto.

Con este objeto se inicia la aplicación DesProc1.0; definiendo o insertando las variables de entrada y de salida, colocando nombres a las mismas o dejando los sugeridos por la herramienta (Ent1, Ent2...para las variables de entrada y Sal1, Sal2,....para las variables de Salida) o cambiando el orden de las variables de entrada.

3.2 Matriz.

EL requisito de operación de este objeto es la existencia del "Gráfico de contexto"; a través del menú "Ver" y el comando "Matriz" se podrá obtener la ventana Matriz. El objeto matriz define las relaciones existentes entre las variables de entrada y una o más variables de salida. Cuando el objeto se activa aparece una matriz de **n** filas, que representan las variables de entrada, y **m** columnas, que representan las variables de salida. En los encabezados de cada una de las filas y columnas aparecerán seis letras iniciales de cada variable. Para establecer las relaciones entre las variables de entrada y salida, solo es necesario hacer un clic[10] en el 'Mouse', en la casilla de intersección entre la entradas y las salidas que se

[10] El proceso es igual cuando se desee desactivar alguna relación que se haya colocado con anterioridad.

deseen relacionar. Este objeto (ventana) una vez activo, no podrá cerrarse, porqué es fundamental en la descomposición.

3.3. Ventana Sentencias Iniciales.

Definidas las variables que intervienen en el proceso y las relaciones entre las mismas, es posible visualizar las relaciones en forma de sentencias. Es indispensable que todas las variables de entrada se encuentren relacionadas al menos con una variable de salida. Este objeto, activo, tiene la función de representar algebraicamente las relaciones en forma interna o externa (vista para el usuario). Esta ventana u objeto, a diferencia de Matriz y Gráfico de Contexto, puede cerrarse o activarse en el momento en que se requiera; ella refleja cualquier cambio realizado en "Gráfico de Contexto" y "Matriz".

3.4. Descomposición en Sentencias.

El objeto que lleva a cabo el proceso de descomposición requiere de la existencia de matriz y grafo de contexto. El resultado de dicho proceso visualiza el producto en dos formas: sentencias (representación algebraica de la descomposición) y grafos de Diagramas de Flujo de Datos -DFD- (Representación de usuario, que muestra las variables con los seis primeros caracteres de los nombres asignados por el objeto "Gráfico de Contexto").

3.5. Descomposición en DFD.

Este objeto muestra la otra forma de representar la descomposición a través de los DFD's. Su representación gráfica utiliza las siguientes convenciones: Los círculos azules representan cada uno de los subprocesos en los cuales se dividió el proceso inicial. Las flecha representan flujos de datos entre procesos. Las letras expresan la representación interna de las variables.

3.6. Interpretación Sentencias.

Al igual que la descomposición, la Interpretación puede representarse de dos formas; sentencias o matrices. Las Sentencias son la representación algebraica de las relaciones resultantes de la descomposición del proceso; tienen una representación interna y representación externa. La representación interna muestra las variables de entrada y de salida con las letras asociadas según lo haya hecho el objeto "Gráfico de Contexto". La representación externa muestra las variables con los seis primeros caracteres de los nombres asignados en el "Gráfico de Contexto". Este objeto puede abrirse o cerrarse en cualquier momento.

3.7. Interpretación y Representación Matriz.

Este objeto presenta una Matriz con las mismas características de "Matriz". Las relaciones originales se representan con los círculos de color azul, mientras que las relaciones adicionales se encuentran representadas por medio de los círculos de color rojo. El objeto puede desactivarse en el momento que se requiera y activarse siempre y cuando, todas las variables de entrada tengan alguna relación con por lo menos una variable de salida.

3.8. Ayudas.

El objeto ayudas puede invocarse con el comando "Contenido" del menú "Ayuda" o a través de su icono correspondiente en al barra de herramientas. La ventana principal de ayudas, presenta los temas teóricos principales de la fundamentación y de la herramienta en sí misma. Se maneja a través de hipertextos o recuadros explicativos o a través de temas claves específicos.

4. PRUEBAS

Para la validación de la herramienta Software desarrollada se realizó un generador automático de pruebas y cerca de 10 pruebas funcionales con problemas de procesos reales. Para el caso de las pruebas funcionales presentamos un ejemplo (Tabla No. 1):

Tabla No. 1. Prueba funcionamiento No. 6. "Producción de Bienes Industriales".

Evaluador	Jaime Camacho
Carrera	Ingeniería Industrial
Facultad	Físico Mecánicas
Tema o Proceso a Analizar	Producción de Bienes Industriales

Descripción del proceso: En la producción de bienes industriales[4] se deben elaborar productos terminados con determinados estándares de calidad, tiempo y especificaciones técnicas. Las materias primas, el tipo de tecnología (maquinaria - equipo - procesos), la calificación del personal que interviene en el proceso, la gestión administrativa y el marco normativo y legal en que opera la empresa, son variables que tienen incidencia en el producto final.

Descripción de la descomposición: Este problema consta de siete variables de entrada y cuatro variables de salida. Las relaciones definidas entre las variables de entrada y de salida pueden verse en las siguientes figuras.

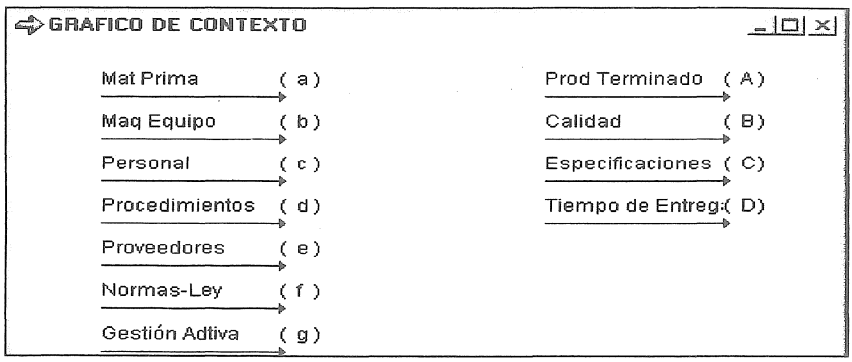


Figura 5. Gráfico de Contexto. "Producción de Bienes Industriales"

Figura 6. Matriz. Producción de bienes Industriales

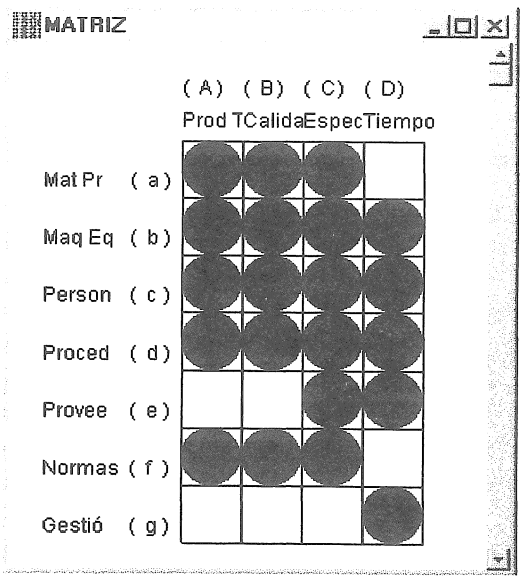


Figura 7. DFD. Producción de bienes Industriales

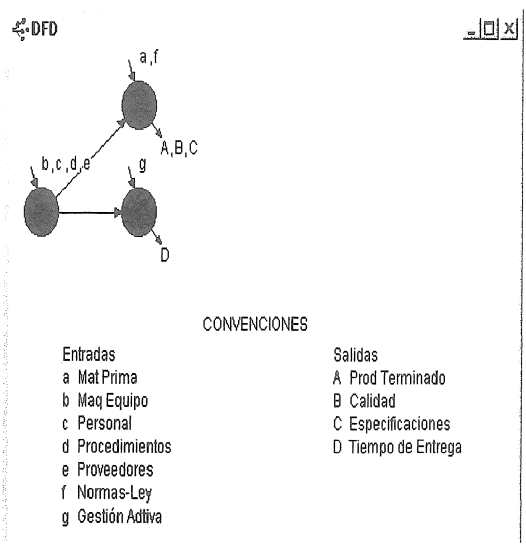
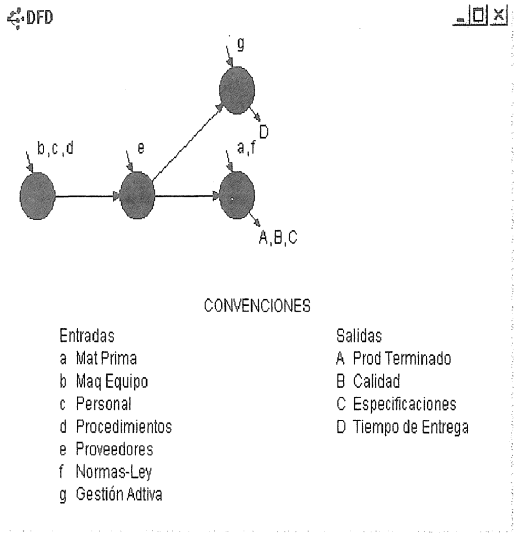


Figura 8. DFD. Producción de Bienes Industriales



El resultado de la descomposición es una sentencia minimal; la calidad de la descomposición medida en la matriz de Interpretación se cataloga como factible. Al igual que en algunos casos anteriores, aparecen relaciones que no se tenían en cuenta en el problema original como lo son (Proveedor -> Producto terminado) y (Proveedor -> Calidad). Si el analista considera que cualquiera de las dos relaciones se puede dar, se conserva la minimalidad y la factibilidad de la misma. Ninguna de las dos relaciones afecta el DFD o la estructura de los nodos subprocesos. Si se consideran las dos relaciones al mismo tiempo, la estructura del DFD cambia, disminuyendo la cantidad de nodos subproceso. Esto último conserva la minimalidad, y la calidad de la misma pasará de factible a óptima. Estos cambios se hacen siempre y cuando el analista los considere necesarios, teniendo en cuenta que estas relaciones no cambian la esencia del problema original. Según el DFD del problema original, los factores como maquinaria y equipo, personal, proveedores, procedimientos y normas de ley, afectan de manera indirecta el tiempo de entrega; el factor primordial para que un producto se entregue a tiempo es la gestión administrativa. Igualmente ocurre con la materia prima; posee determinadas especificaciones y es el factor más importante para que un producto terminado sea de óptima calidad.

5. CONCLUSIONES

5.1. A Nivel de Usuario.

- La rapidez con que se obtienen los resultados y la posibilidad de verlos simultáneamente en diferentes formas, facilita el proceso de prueba de diferentes definiciones para un mismo problema.
- Los usuarios que deseen profundizar sobre la forma en que opera el álgebra, encontrarán en la herramienta un excelente apoyo para indagar sobre la posibilidad obtener mejoras a la especificación teórica utilizada.

5.2. A Nivel de Desarrollo.

- La formalización proveída para el proceso de descomposición, elimina la subjetividad en la especificación de procesos; probar el hecho es ahora posible, obteniéndose descomposiciones "lógicas" para que los expertos consultados, en los casos analizados, los evalúen.
- La definición de un problema de flujos de datos mediante relaciones entre variables, sin considerar el grado de incidencia de unas sobre otras, sino la simple certeza de la influencia entre ellas, permite descubrir flujos que habían sido considerados de poca importancia. En algunos casos se llegan a descubrir incidencias desconocidas de una variable sobre otra.
- Los operadores del álgebra de Adler, en la casi totalidad de los casos, pueden aplicarse en un orden fijo; la aplicación sucesiva de un mismo operador sobre un lista de transformadas hace que éstas lleguen a un estado en el que ya no es posible aplicar dicho operador, y por lo tanto la única forma de continuar con el proceso de descomposición, es la prueba con otros operadores.
- El hecho de aplicar los operadores del álgebra de Adler, en diferente orden, conlleva[11] a la obtención de descomposiciones muy distintas.
- La medida de la calidad o interpretación de la descomposición es indicio de relaciones adicionales con respecto a las relaciones originales.
- Puesto que DesProc 1.0, formaliza el proceso de descomposición, los resultados serán, mas o menos lógicos, en la medida que sean mas acertadas las relaciones definidas entre las variables de entrada y las variables de salida; El hecho supone un buen conocimiento de las razones por las cuales se incluye cada variable de entrada o cada variable de salida en el proceso en estudio.

5.3. A Nivel de Aporte.

- La herramienta obtenida permite analizar la descomposición de cualquier proceso independientemente de su naturaleza; no tiene en cuenta las características físicas u abstractas representadas por cada variable.
- Dado que el álgebra de Adler no considera la posibilidad de llegar a sentencias "no minimales" no provee mecanismos para solucionar este problema; razón necesaria definir algunas políticas adicionales alternas para suplir esta deficiencia.

[11] Aunque en la mayoría de los casos las diferencias son mínimas

- DesProc 1.0, se concibe como eslabón o apoyo en la cadena de aplicaciones que puedan beneficiarse durante el proceso de especificación de productos software.
- La incidencia del orden de las variables de entrada en el resultado de la descomposición obtenida permitió desarrollar un mecanismo de análisis para interpretar dicha incidencia.
- El estudio del álgebra nos llevó a cuestionar el funcionamiento de la misma, haciendo algunas conjeturas acerca de los operadores; la aparición de un solo subproceso como el inicial o la inexistencia de ciclos en la descomposición.

6. Bibliografía

[1] Addler M. *"An Algebra for Data Flow Diagram Process Decomposition"*. Transaction on Software Engineering. Vol. 14 #2 February de 1988.

[2] Llamosa-Villalba R. "Specifications Systems Information- (SSI). Specifications and Requirements for Systems Information Model", XIV IASTED International Conference, Applied Informatics, Innsbruck, Austria. February 20-22, 1996.

[3] Llamosa-Villalba R., *Evaluation and Synthesis of Computational Methods, Formalisms and Tools in order to develop Information Systems Specification,* System and Computer Science Engineering School, UIS, Bucaramanga, Colombia, February, 1996

[4] Harrinton H.J. *Mejoramiento de los procesos de la empresa*, Santafé de Bogotá, MacGraw Hill, 1992, 96p.

[5] Protecko et all, *"Towards the automatic generator of Software Diagrams"*, *IEEE Transactions on Software Engineering*, Vol. 17 No. 1, January 1991, 10-21p..

[6] Arndt T. and Gersoy Angela, *"The Composition of data flow"*, Proc. 4 4, int Conf. Software eng. Knowledge Eng. Public by *IEEE Computer Society, Los Alamitos, CA, USA*, 1992, 560-566p.

UN ALGORITMO PARA REESTRUCTURAR PROGRAMAS PROCEDURALES

Cobo, Hernán Mauco, María Virginia

Instituto de Sistemas de Tandil

Facultad de Ciencias Exactas

Universidad Nacional del Centro de la Pcia. de Buenos Aires

San Martín 57 - (7000) Tandil - Bs. As. - Argentina

e-mail: hcobo@tandil.edu.ar vmauco@tandil.edu.ar

Resumen

Un aspecto importante en el mantenimiento de software es la existencia de programas viejos que son difíciles de comprender y mantener porque son grandes y están desestructurados.

En este artículo se presenta la primera parte de un proyecto para reestructurar programas escritos en algún lenguaje procedural (Cobol, Pascal, C, etc). El objetivo de esta etapa ha sido, además de la reestructuración sintáctica de los mismos, la definición de una representación que no sólo permitiera almacenar toda la información contenida en el código fuente de un programa sino que también contribuyera a la simplicidad de los algoritmos definidos para transformarla.

El artículo describe el algoritmo de reestructuración desarrollado y muestra que el grafo de dependencias del programa extendido, que es la representación interna definida, es una estructura adecuada para resolver este problema.

Palabras claves: reestructuración de código, dependencias de control, mantenimiento de software, grafo de dependencias del programa, reingeniería.

1. INTRODUCCION

Una gran cantidad del software existente tiene una media de edad de entre 10 y 15 años (Pressman, 1993). Aunque esos programas se hubieran desarrollado con las mejores técnicas de diseño y codificación existentes en su momento, luego migraron a plataformas nuevas, se ajustaron a cambios en el hardware y en los sistemas operativos y se mejoraron para satisfacer las necesidades de nuevos usuarios. El resultado de esto es la existencia de sistemas de software con estructuras de datos pobremente diseñadas, una codificación, una lógica y una documentación pobres, que tienen que seguir funcionando en la actualidad.

El mantenimiento de software se lleva gran parte del presupuesto destinado a un proyecto de desarrollo de software (Pressman, 1993). Por esto, en los últimos años las herramientas de ingeniería reversa y reingeniería están empezando a ocupar un lugar fundamental en el proceso de mantenimiento de software. La ingeniería reversa extrae información de diseño utilizando solamente el código fuente de un programa. La reingeniería toma la información obtenida y reestructura el programa para mejorar su calidad, y por lo tanto simplificar su mantenimiento. El software resultante de la reingeniería reimplementa la función del sistema existente, aunque también se pueden añadir nuevas funciones.

Los cambios continuos tienden a degradar la estructura del software generando costos de mantenimiento más elevados (Griswold et al., 1992). En algunos casos entonces, vale la pena manipular la estructura de un sistema para hacer que los cambios sean más fáciles. Dentro de estas manipulaciones estructurales, las técnicas de reestructuración de código ocupan un lugar muy importante.

El objetivo de la reestructuración es transformar programas no estructurados en programas estructurados equivalentes que se puedan expresar en términos de secuencias y anidamientos de programas más pequeños con una entrada y una salida, y que se puedan leer y entender sistemáticamente (Arnold, 1993).

Sin embargo, la reestructuración manual es una actividad cara y propensa a errores. Por eso, el gran potencial de la reestructuración automática de programas está en manejar la estructura de programas grandes, en la escala de millones de líneas de código, donde hay un gran impacto económico.

En este trabajo se presenta un algoritmo para reestructurar programas escritos en un lenguaje procedural como Cobol, Pascal, C, etc. La reestructuración sintáctica constituye la primera etapa de un proyecto cuyo objetivo final es lograr agrupar todo el conocimiento necesario para obtener, desde un programa fuente, una estructura de más alto

nivel que se pueda analizar y manipular de forma tal que el nuevo código generado a partir de esta estructura, además de preservar la funcionalidad original, sea estructurado, más legible, modular, flexible, reusable, etc.

Se describe además el grafo de dependencias del programa extendido que es la representación interna definida para almacenar y manipular la información extraída del código fuente, y se muestra que mediante transformaciones simples es posible convertir un programa no estructurado en su equivalente estructurado, conservando la funcionalidad del programa original.

2. BENEFICIOS DE LA REESTRUCTURACION

La complejidad de un programa se puede correlacionar, en parte, con la complejidad topológica de su grafo de flujo de control (Bush, 1985). Los programas no estructurados tienen grafos cuyos nodos están tan conectados uno con otro que el grafo no se puede particionar en regiones independientes. Esto implica que los programas se pueden entender solamente como un todo. Por esto, el primer paso para contribuir a la comprensión de los mismos es transformarlos de modo que la lógica del programa y el texto se correspondan precisamente.

El objetivo de la reestructuración es enmendar los programas que han sufrido las cicatrices de decenas de cambios apresurados realizados por imperio de las circunstancias, y que los han convertido en componentes inestables, poco fiables y plagadas de defectos sutiles que sólo aparecen en los momentos menos deseables (Boria, 1992). La reestructuración comprende el proceso de reorganización de la lógica procedural de un programa de modo que cumpla las reglas de la programación estructurada; la reestructuración automática es la traducción mecánica de un programa (no estructurado) en un programa estructurado funcionalmente equivalente.

La programación estructurada es una forma normal sobre los grafos de flujo de control de los programas (Bush, 1985). Un grafo de flujo de control está en forma normal estructurada, si está compuesto solamente de tres tipos de subgrafos simples: la secuencia, el condicional y la iteración, cada uno de los cuales tiene un solo punto de entrada y un solo punto de salida. Por eso, un grafo construido a partir de ellos se puede factorizar en una jerarquía top-down de muchos grafos más pequeños.

Entre los beneficios de la forma normal estructurada (Bush, 1985) quizás el más obvio sea mejorar la comprensión del programa. Dividiendo un grafo en lotes de pequeñas partes auto-contenidas, el código fuente del programa resultante se divide en una cantidad de pequeños módulos auto-contenidos, dando lugar a un programa que tiene

muchas vistas pequeñas, en oposición a un programa no estructurado que tiene sólo una vista grande. Otro beneficio surge cuando se necesita reemplazar o mejorar alguna sección del programa, ya que frecuentemente es posible localizar el cambio a un módulo en particular. También el testing, el debugging y el control de efectos laterales se simplifican considerablemente al realizarse sobre programas estructurados.

3. ALGORITMOS DE REESTRUCTURACION

Los algoritmos de reestructuración de código se pueden dividir en dos grupos: a)- el proceso de reestructuración implica modificar el programa original agregando una o más variables de control que permiten simular estructuración; b)- el proceso de reestructuración involucra el reconocimiento y reemplazo de porciones de código no estructurado por clichés (porciones de código que implementan una estructura de datos o un algoritmo en particular).

Entre los algoritmos del grupo a) se pueden mencionar el de Böhm y Jacopini (Böhm et al., 1966) y el de Linger, Mills y Witt (Linger et al., 1979). Böhm y Jacopini mostraron que todo diagrama de flujo de programa se puede traducir en un programa while equivalente (con una sentencia while), introduciendo nuevas variables lógicas, nuevos predicados para evaluar estas variables, junto con asignaciones para hacer que su valor sea verdadero o falso. Esta transformación es indeseable ya que cambia totalmente la topología del programa, aún cuando el mismo estuviera ya estructurado, dando como resultado un programa que es más difícil de entender. El algoritmo propuesto por Linger, Mills y Witt, agrega una sola variable que actúa como contador de programa, y aunque preserva las partes que tienen estructura aceptable, modifica la estructura general del programa.

Los algoritmos clasificados dentro del grupo b), como el algoritmo propuesto por Bush (Bush, ,1985), trabajan en forma bottom-up y, a partir de una biblioteca de clichés, van armando jerarquías hasta que se ha reestructurado todo el programa o no hay clichés definidos para las porciones de código que quedan. Algunos de los problemas de este enfoque (Rich et al., 1990), son la definición y actualización de una biblioteca de clichés que contenga la mayor cantidad de casos posibles, la existencia de código no reconocible y la limitación de su aplicación a programas de un determinado dominio. Por lo tanto, la solución que proponen este tipo de algoritmos no es completa.

4. DEFINICION DE UNA ESTRUCTURA PARA REPRESENTAR LA INFORMACION DEL PROGRAMA

La definición de una representación interna para los programas constituyó uno de los aspectos más importantes de este trabajo. La relevancia de esta decisión se fundamenta en que dicha representación no sólo tendría que almacenar toda la información posible extraída del código fuente de un programa, en lo que respecta a flujo de datos y de control, sino también en que la complejidad de los algoritmos definidos para transformarla estaría afectada en gran medida por la facilidad o no de acceso y manipulación de la información almacenada en la estructura. Además, el éxito del método de reestructuración dependería directamente de que la información obtenida y almacenada fuera correcta y completa.

La estructura definida es el grafo de dependencias del programa extendido que permite representar las dependencias de control y de datos para cada operación del programa, y que corresponde a una extensión del grafo de dependencias del programa (GDP) propuesto en (Ferrante et al., 1987). Aunque el desarrollo del GDP tuvo su origen en aplicaciones relacionadas con optimización de compiladores, trabajos posteriores demostraron las ventajas del uso de esta representación en ingeniería de software (Horwitz et al., 1988; Horwitz et al., 1992).

4.1. GRAFO DE DEPENDENCIAS DEL PROGRAMA (GDP)

La representación correspondiente al GDP (Ferrante et al., 1987) es un grafo orientado en el cual los nodos son sentencias y expresiones de predicados y los arcos incidentes a un nodo representan los valores de datos y las condiciones de control de las cuales dependen las operaciones del nodo. El conjunto de todas las dependencias de un programa induce un orden parcial sobre las sentencias y predicados en el programa; dicho orden se debe seguir para preservar la funcionalidad del programa original.

Las dependencias son de dos tipos: **dependencias de datos**, que surgen entre dos sentencias cuando una variable que aparece en una de ellas podría tener un valor incorrecto si las dos sentencias se invirtieran, y **dependencias de control**, que se dan entre una sentencia y el predicado cuyo valor controla inmediatamente la ejecución de esa sentencia.

Los arcos del GDP representan dependencias de control o de datos entre componentes del programa. Los arcos de dependencias de control están rotulados T o F, y la fuente de un arco de este tipo es siempre el nodo Entry o un nodo predicado. Un arco de dependencia de control de un nodo n_1 a un nodo n_2, $n_1 \rightarrow_R n_2$, indica que durante la ejecución,

siempre que se evalúa el predicado representado por n_1 y su valor coincide con el rótulo R sobre el arco a n_2, se ejecutará la componente del programa representada por n_2. Existen dos clases de arcos de dependencias de datos (Binkley et al., 1995): arcos de dependencia por flujo y arcos de dependencia def-order. Un arco de dependencia por flujo $v \rightarrow_f w$, va de un nodo v, que representa una asignación a una variable x, a un nodo w que representa un uso de x alcanzado por esa asignación. Un arco def-order va desde un nodo v a un nodo w si ambos contienen una definición de la misma variable x, las dos definiciones alcanzan un uso común de x en un nodo u, es decir $v \rightarrow_f u$ y $w \rightarrow_f u$, y v precede léxicamente a w.

4.2. GRAFO DE DEPENDENCIAS DEL PROGRAMA EXTENDIDO (GDPE)

Como la primera parte de este proyecto apunta a lograr la reestructuración de programas en lo que respecta a su estructura sintáctica, sólo será necesario tener en cuenta las dependencias de control.

El punto de partida para la construcción del GDPE correspondiente a las dependencias de control, es el grafo de flujo de control de un programa, aumentado con un nodo Inicio (representa la entrada al programa) y un nodo Fin (en él confluyen todas las salidas del programa). El grafo de flujo de control es un grafo dirigido en el cual cada nodo representa una sentencia y cada arco representa el flujo de control entre sentencias. Los arcos que representan transferencia de control condicional tienen rótulo T (true) o F (falso); los demás arcos son no rotulados. En (Ferrante et al. 1987) se puede encontrar un método para determinar dependencias de control en programas arbitrarios. En este trabajo se realizó una extensión a la estructura construida por este algoritmo, que consiste en considerar el nivel de cada uno de los nodos .

En general, se define nivel de un nodo n como la longitud del camino que comienza en el nodo Entry y termina en el nodo n. Cuando un nodo n depende de varios nodos, y por lo tanto existen varios caminos hasta él desde el nodo Entry, se define su nivel como la longitud del camino más largo desde el nodo Entry al nodo n, tal que n no es un nodo interior en ese camino. Se considera que el nivel 0, correspondiente al nodo Entry, es el nivel superior.

La importancia de la incorporación del nivel para cada uno de los nodos del GDPE está en que permite determinar rápida y simplemente si un programa está o no estructurado, evitando tener que recorrer el GDPE para detectar ciclos. Además, contribuye a simplificar las transformaciones definidas para convertir un programa no estructurado en uno estructurado.

Por ejemplo, en el GDPE de la Figura 5, el nivel de los nodos 2 y 12 es 4 y 3, respectivamente.

Una de las diferencias más importantes del GDPE con el grafo de flujo de control es que expone paralelismo potencial. Esto significa que dos nodos n_1 y n_2 se podrían ejecutar simultáneamente a menos que existiera una dependencia de datos entre ellos; más aún, se podría intercambiar el orden de ejecución de los nodos sin alterar la funcionalidad del programa original.

Como en esta etapa no se tienen en cuenta las dependencias de datos, se agregan temporariamente arcos que indican el orden de ejecución de los nodos que tienen el mismo nivel, para asegurar la preservación de la funcionalidad del programa original. Este orden se corresponde con el orden impuesto por el grafo de flujo de control.

5. PROGRAMA ESTRUCTURADO Y GDPE ESTRUCTURADO

Tal como se estableció en la sección 2, un programa está estructurado si se construye únicamente a partir de las estructuras de control: secuencia, selección (IF THEN ELSE, IF THEN) e iteración (WHILE DO). En las Figuras 1, 2 y 3 se muestran el grafo de flujo de control aumentado y el GDPE asociado a cada una de estas estructuras.

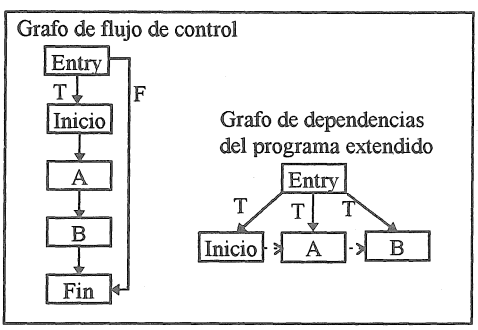


Figura 1: Secuencia

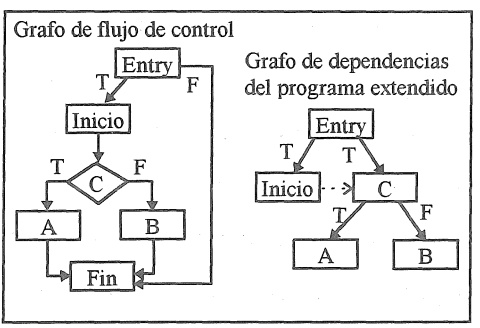


Figura 2: Selección

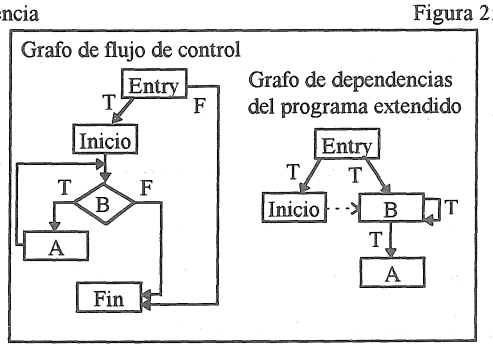


Figura 3: Iteración

Se puede decir entonces que un GDPE, y por lo tanto el programa asociado al mismo, está estructurado si se construye a partir de la composición de los GDPE's correspondientes a las estructuras de control permitidas.

En un GDPE estructurado cada nodo depende por control de solamente uno de los nodos del nivel inmediato superior. Esto implica que el subgrafo inducido por las dependencias de control es un árbol.

Por lo tanto, será necesario y suficiente encontrar un arco de control que destruya la estructura de árbol para determinar que un GDPE, y en consecuencia el programa asociado al mismo, no está estructurado. Esto es equivalente a encontrar un nodo que depende de dos o más nodos.

Se debe notar que para determinar si un GDPE está o no estructurado, no es necesario tener en cuenta los arcos de un nodo a sí mismo, como el arco con rótulo T sobre el nodo B en la Figura 3, ya que la presencia o ausencia de este tipo de arcos sobre un nodo predicado sólo permite determinar si el subgrafo que depende de ese nodo corresponde a una iteración o a una selección respectivamente.

6. ALGORITMO DE REESTRUCTURACION PROPUESTO

Como se mencionó en la sección anterior, es suficiente encontrar un nodo que depende de dos ó más nodos para determinar que un GDPE no está estructurado. Pero haciendo un análisis más detallado de las dependencias de control entre los nodos, se puede establecer que existen dos clases de dependencias, claramente diferenciadas entre sí, que implican desestructuración en el GDPE, dando lugar a los siguientes tipos de desestructuración:

Tipo A) al menos un nodo depende de dos o más nodos de niveles superiores;

Tipo B) al menos un nodo depende de nodos de niveles inferiores, determinando la aparición de un ciclo en el GDPE.

Por eso, las transformaciones posibles a aplicar al GDPE serán dos, una correspondiente a cada tipo. Sin embargo, el objetivo de ambas es el mismo: convertir el subgrafo inducido por las dependencias de control del GDPE en un árbol.

Es importante destacar que los subgrafos del GDPE que no contengan dependencias del **tipo A** ó **B**, es decir que ya estén estructurados, no sufrirán ninguna modificación en el proceso de reestructuración. Por lo tanto, la aplicación del algoritmo de reestructuración a un programa ya estructurado dará como resultado el programa original.

El proceso de reestructuración elimina primero las dependencias de **tipo A**. Comenzando con los nodos de nivel inferior y para cada nodo n que depende de dos o más nodos de niveles superiores n_1, n_2, ..., n_k, se hacen k copias del

subgrafo que tiene a n como nodo inicial y que está constituido por nodos de nivel inferior a n, cada uno con todos sus arcos emergentes; cada copia se hace depender de n_1, n_2, ..., n_k, con el rótulo correspondiente. Esto significa que para aplicar esta transformación a un nodo se deben eliminar, previamente, las dependencias de tipo A para todos los nodos de nivel inferior que dependen de él. Se debe destacar que, aún cuando los nodos a duplicar se eligieran en orden arbitrario, el resultado final de estas transformaciones es el mismo. La razón por la cual se aplican desde los nodos de nivel inferior del GDPE es que de esta forma se puede asegurar la consistencia de los GDPE's intermedios.

Para eliminar los ciclos en el GDPE se tratan las dependencias de tipo B. La existencia de un ciclo en el GDPE significa que el control puede pasar más de una vez por el conjunto de nodos involucrados en el ciclo, y esto se reflejará en el código estructurado en la aparición de una estructura iterativa que no estaba presente en el código original. Por eso, la transformación correspondiente implicará agregar una variable lógica para controlar dicho ciclo.

Para cada ciclo los pasos a seguir son los siguientes:

- detectar el ciclo; esto implica encontrar en el GDPE un arco de control para el cual el nivel del nodo destino n_d es superior al nivel del nodo fuente n_f;
- insertar en el mismo nivel de n_d dos nuevos nodos: 1) un nodo con una asignación T a una variable lógica L, cuyo nombre puede ser determinado por el usuario; 2) un nodo predicado n_p , c}ya condición es que el valor de L sea T, con un arco de control a sí mismo rotulado T; los dos nodos agregados dependerán por control y con el mismo rótulo, del nodo de nivel superior del que dependía el nodo n_d ;
- dependiendo del nodo predicado n_p y con rótulo T, agregar un nodo con una asignación F a la variable lógica L;
- eliminar la dependencia de n_d del nodo de nivel superior, haciendo que n_d dependa por control y con rótulo T del nodo predicado n_p ;
- reemplazar cada dependencia por control con destino n_d y cuyos nodos fuente tengan nivel inferior a n_d , por una dependencia por control con el mismo rótulo que tenga como destino un nuevo nodo con una asignación T a la variable lógica L.

A diferencia de otros algoritmos analizados, el algoritmo desarrollado sólo agrega variables cuando las dependencias de control del programa determinan que el GDPE correspondiente contenga al menos un ciclo.

Las Figuras 4 y 5 contienen parte del texto de un programa escrito en Cobol y el grafo de dependencias del programa extendido correspondiente, respectivamente. Como puede observarse el GDPE no es un árbol; esto significa que el

programa asociado no está estructurado. La aplicación del algoritmo de reestructuración diseñado permite convertirlo en un árbol (Fig. 6), a partir del cual se deriva el código del programa estructurado correspondiente (Fig. 7).

Tanto en el GDPE desestructurado como en el estructurado, el contenido de cada nodo es el siguiente (teniendo en cuenta el tipo de cada uno, sentencia o predicado):

1: SECCION=3
2: READ EMP NEXT RECORD
3: CLOSE EMP
4: MOVE ZERO TO EXTRA
5: HORAS=160
6: CATEGORIA=1
7: CATEGORIA=0
8: EXTRA>0
9: COMPUTE SUEL=160*VALOR_HS+EXTRA
10: COMPUTE SUEL=160*VALOR_HS
11: MOVE SUEL TO SUELDO
12: HORAS>160
13: COMPUTE EXTRA= (HORAS-160)*VAL_EX

```
PROC_SUELDO_EMP.
      IF SECCION = 3 GOTO SALIDA
      ELSE GOTO EMPLEADOS.
HRS_EXTRAS.
      COMPUTE EXTRA=(HORAS-160)*VAL_EX.
      GOTO PROC_SUELDO.
EMPLEADOS.
       READ EMP NEXT RECORD.
       MOVE ZERO TO EXTRA.
       IF HORAS=160 GOTO PROC_SUELDO.
       IF HORAS>160 GOTO HRS_EXTRAS
       ELSE GOTO PROC_SUELDO_EMP.
PROC_SUELDO.
      IF CATEGORIA=1 GOTO EMPLEADOS
      ELSE IF CATEGORIA=0 GOTO SALIDA
      ELSE GOTO VALOR_SUELDO.
SALIDA.
     CLOSE EMP.
      STOP RUN.
VALOR_SUELDO.
      IF EXTRA>0
          COMPUTE SUEL=160*VALOR_HS+EXTRA
      ELSE
          COMPUTE SUEL=160*VALOR_HS.
      MOVE SUEL TO SUELDO.
      GOTO PROC_SUELDO_EMP.
```

Figura 4: Texto Cobol desestructurado

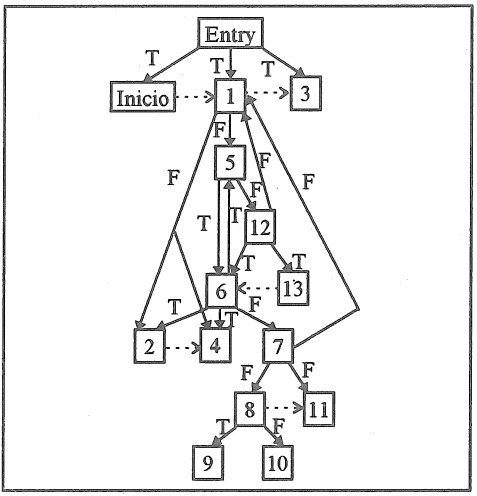


Figura 5: Grafo de dependencias del programa extendido, desestructurado

Los dos tipos de desestructuración planteados en el punto anterior, se corresponden con determinadas características del código fuente del programa. El **tipo A** corresponde a sentencias GOTO "hacia adelante". La solución a este tipo de desestructuración implica duplicación de código. El algoritmo detecta esta duplicación y codifica el conjunto de sentencias duplicadas como un párrafo, permitiendo que el usuario le asigne un nombre. El **tipo B** corresponde a sentencias GOTO "hacia atrás", que determinan la aparición de ciclos en el GDPE. Por eso, la solución implica el agregado de variables, para controlar esos ciclos. Pero como estas variables tienen normalmente una semántica

dentro del programa, se da al usuario la posibilidad de ponerles un nombre para que el programa estructurado resultante sea más legible. Por ejemplo, a las variables L1 y L2 de la Fig. 6 agregadas por el algoritmo de reestructuración, se las puede reemplazar en el código final por CATEG_NO_CERO y CATEG_UNO respectivamente.

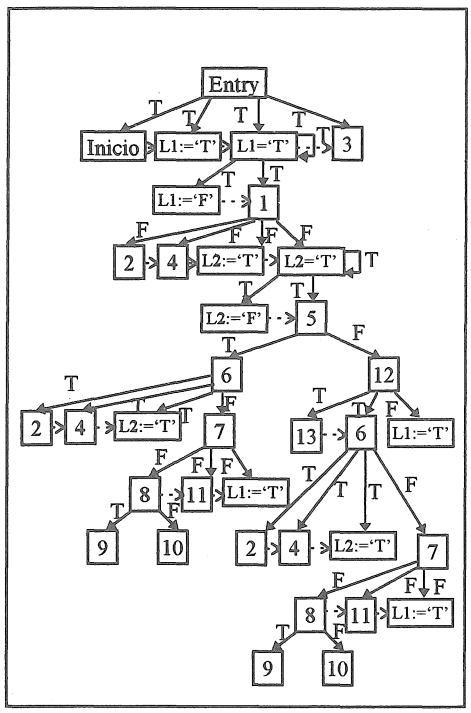


Figura 6: Grafo de dependencias del programa extendido, estructurado

```
PRINCIPAL.
      MOVE 'T' TO CATEG_NO_CERO.
      PERFORM UNTIL CATEG_NO_CERO='F'
        MOVE 'F' TO CATEG_NO_CERO
        IF NOT (SECCION=3)
         READ EMP NEXT RECORD
         MOVE ZERO TO EXTRA
         MOVE 'T' TO CATEG_UNO
         PERFORM UNTIL CATEG_UNO='F'
          MOVE 'F' TO CATEG_UNO
          IF HORAS=160
           PERFORM CALC_SUELDO
          ELSE
          IF HORAS>160
           COMPUTE EXTRA=(HORAS-160)*VAL_EX
           PERFORM CALC_SUELDO
          ELSE
           MOVE 'T' TO CATEG_NO_CERO
          END-IF
          END-IF
         END-PERFORM
        END-IF
      END-PERFORM.
      CLOSE EMP.
      STOP RUN.

CALC_SUELDO.
      IF CATEGORIA=1
        READ EMP NEXT RECORD
        MOVE ZERO TO EXTRA
        MOVE 'T' TO CATEG_UNO
      ELSE
       IF NOT (CATEGORIA=0)
        IF EXTRA>0
         COMPUTE SUEL=160*VALOR_HS+EXTRA
        ELSE
         COMPUTE SUEL=160*VALOR_HS
        END-IF
        MOVE SUEL TO SUELDO
        MOVE 'T' TO CATEG_NO_CERO
       END-IF
      END-IF.
```

Figura 7: Texto Cobol estructurado

7. FUTUROS DESARROLLOS

Como se explicó anteriormente, este trabajo constituye la primera etapa de un proyecto para reestructurar programas escritos en un lenguaje procedural. La próxima etapa del proyecto apunta a transformar un programa sintácticamente estructurado, en un programa modular.

En un programa bien diseñado y poco mantenido, cada funcionalidad del usuario se puede ubicar en un solo módulo. Sin embargo, en código viejo es más probable que la misma esté desparramada por varios módulos o programas, complicando el mantenimiento del mismo.

Por eso, se estudiarán las estructuras y los flujos de datos del programa para desarrollar una técnica que permita reconocer y extraer porciones de código funcionalmente relacionadas para agruparlas en un módulo.

8. CONCLUSIONES

El mantenimiento es la etapa más cara del proceso de desarrollo de software. La estructura de un programa ejerce una gran influencia sobre el costo de mantenimiento.

En este trabajo se presentó un algoritmo para reestructurar sintácticamente programas, que corresponde a la primera etapa de un proyecto cuyo objetivo es mejorar la calidad de los programas haciéndolos estructurados, modulares, reusables, etc. Se mostró además que la simplicidad del algoritmo se debe a la elección del grafo de dependencias del programa extendido como estructura interna para representar la información contenida en un programa.

REFERENCIAS

- Arnold, R., Software Reengineering, 2nd edition, IEEE Computer Society Press, 1994.
- Binkley, D., Horwitz, S., Reps, T., Program Integration for Languages with Procedure Calls, ACM Transactions on Software Engineering and Methodology, Vol. 4, N° 1, 3-35, 1995.
- Böhm, C., Jacopini, G., Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules, Communications of the ACM, Vol. 9, N° 5, 366-371, 1966.
- Boria, J., Mauco, M., Moreno, M., Un Prototipo de Ambiente para la Reestructuración de Programas Cobol, Anales de USUARIA '92, Bs. As., Argentina, 133-143, mayo 1992.
- Bush, E., The Automatic Restructuring of Cobol, Proc. of the Conference on Software Maintenance, 35-41, 1985.
- Ferrante, J., Ottenstein, K., Warren, J., The Program Dependence Graph and its Use in Optimization, ACM Transactions on Programming Languages and Systems, Vol. 9, N° 3, 319-349, 1987.
- Griswold, W., Notkin, D., Automated Assistance for Program Restructuring, Technical Report Number CS92-221, University of California, 35 pages, 1992.
- Horwitz, S., Prins, J, Reps, T., On the Adequacy of Program Dependence Graphs for Representing Programs, 15th ACM Symp. on Principles of Programming Languages, 12 pages, Jan 1988.
- Horwitz, S., Reps, T., The Use of Program Dependence Graphs in Software Engineering, 14th International Conference on Software Engineering, 20 pages, 1992.
- Linger, R., Mills, H., Witt B., Structured Programming: Theory and Practice, Addison-Wesley Publishing Company, Cambridge, Mass., 1979.
- Pressman, R., Ingeniería del Software: Un Enfoque Práctico, 3° edición, Mc Graw Hill, 1993.
- Rich, C., Wills, L., Recognizing a Program's Design, IEEE Software, Jan 1990.

A Utilização de Múltiplos Métodos Formais na Análise de Requisitos de Software

Simone C. dos Santos (scs@di.ufpe.br)
Fabio Q. B. da Silva (fabio@di.ufpe.br)
Departamento de Informática
Universidade Federal de Pernambuco

Sumário

Este artigo propõe a utilização de múltiplos formalismos na análise de requisitos de software para especificar aspectos diferentes de um sistema complexo, explorando o poder de expressão de cada método e considerando a necessidade de relacionamento entre as especificações. Para isto, um sistema de caldeiras (*Boiler System*) é utilizado como estudo de caso, para o qual duas especificações formais são desenvolvidas, constituindo dois modelos diferentes do sistema: consistência de medidas dos dispositivos (descrito em TLA) e transições de estados considerando falhas na comunicação entre subsistemas controladores (descrito em Z). Uma relação matemática rigorosa entre TLA e Z é estabelecida, permitindo a integração entre as duas especificações através de provas de consistência. Os resultados descritos aqui fazem parte do trabalho desenvolvido em [3].

Abstract

This article proposes the use of multiple formalisms in the software requirements analysis to describe distinct aspects of a complex system, exploring the expressive power of each method and considering the relationship between the specifications. For this purpose, a boiler system is used as a case study, in which two formal specifications are developed, building two different models of the system: consistency of the device's measures (described using TLA) and state transitions taking into account communication failures between the controlling subsystems (described using Z). A rigorous mathematical relation is established between Z structures and TLA structures, allowing the integration of the specifications via formal proofs of consistency. The results reported here are part of the work developed in [3].

1 Introdução

A análise e especificação de requisitos representa o primeiro estágio do modelo clássico do ciclo de vida de software. O primeiro passo a ser tomado pelo engenheiro de software neste estágio é entender o problema que se deseja resolver possibilitando a verificação de que o sistema a ser construído atende aos objetivos estabelecidos.

Considerando que o engenheiro de software conhece pouco sobre o sistema, torna-se fundamental aumentar a precisão dos requisitos descritos nesta fase, especificando o que o sistema deve fazer e omitindo detalhes de como deve ser feito através de uma linguagem matemática precisa e não ambígua, isto é, formal. Ao especificar formalmente um sistema, o engenheiro de software é obrigado a entender melhor o problema, descrevendo o sistema de forma precisa e clara, com a possibilidade de provar matematicamente certas propriedades que devem ser satisfeitas pelo sistema.

Considerando o desenvolvimento de sistemas complexos, os requisitos iniciais podem ser considerados sob vários aspectos diferentes, por exemplo: comportamento determinístico, concorrente ou temporal. Desta forma, a especificação de requisitos pode ser dividida em subfases menores ao considerar cada um destes aspectos separadamente, reduzindo a quantidade de detalhes a serem considerados por vez e, com isso, proporcionando um maior entendimento e facilidade para explorar as características de cada um destes aspectos.

Entretanto, a capacidade de descrever todos estes aspectos geralmente não é encontrada em um único método formal. Embora algumas linguagens possam descrever mais de um aspecto, é difícil encontrar um formalismo capaz de tratar com a mesma facilidade todas as características de um sistema complexo. Por exemplo, a linguagem MaMOOZ [5] permite expressar considerações de tempo além de aspectos deterministas, mas não modela concorrência.

Desta forma, a abordagem proposta por este artigo utiliza formalismos diferentes para especificar cada aspecto do sistema, explorando o poder de expressão de cada método e considerando a necessidade de relacionamento entre as várias especificações.

Para que as especificações possam co-existir no mesmo projeto, elas devem ser consistentes entre si. Desta forma, é necessário provar matematicamente alguma relação que defina esta consistência. A integração de tais especificações é realizada através de provas de consistência que preservam as propriedades de segurança satisfeitas por cada uma delas separadamente. Desta forma, uma especificação consistente, mais completa e confiável é obtida.

Neste artigo, apresentaremos o enfoque discutido acima através de um estudo de caso, um sistema crítico de controle de caldeiras, denominado ao longo do texto por *Boiler System*. A linguagem TLA (*Temporal Logic of Actions* [6]) é utilizada para descrever as propriedades de segurança deste sistema. As características funcionais são especificadas utilizando a linguagem Z [10]. As justificativas e motivações para a escolha de tais métodos ficam claras no decorrer do texto.

Para tornar mais clara a compreensão dos conceitos, vários detalhes formais foram omitidos neste artigo. O leitor interessado nos detalhes técnicos e na abordagem mais teórica deste problema é convidado a ler [3].

O trabalho é organizado em 6 seções. A Seção 2 descreve rápida e informalmente o sistema de caldeiras que será usado como estudo de caso. As Seções 3 e 4 apresentam fragmentos da especificação formal do sistema descritos em TLA e Z. A Seção 5 mostra como provar a consistência entre as duas especificações e, por fim, a Seção 6 apresenta as conclusões e considerações finais.

2 Estudo de Caso: um Sistema de Caldeiras

Esta seção descreve um sistema de caldeiras utilizado como estudo de caso para abordagem proposta por este artigo, referenciado ao longo do texto por *Boiler System*. Aspectos importantes referentes

às propriedades de segurança e funcionalidade deste sistema serão especificadas posteriormente utilizando TLA e Z, respectivamente. A escolha deste exemplo é justificada pela inerente complexidade deste sistema considerado de segurança crítica, onde características como concorrência, transição de estados e propriedades temporais precisam ser consideradas e descritas.

A descrição de tal sistema foi proposta no Simpósio Internacional de Projeto e Sistemas de Software Relacionados a Segurança (Waterloo, Canadá, [8]), com a finalidade de promover uma competição entre os participantes através do enfoque de desenvolvimento formal aplicado a um problema genérico.

2.1 O *Boiler System*

O *Boiler System* é composto por um vasilhame aquecido, produzindo vapor d'água saturado através de um duto no topo do vasilhame e cujo fluxo de água em seu interior é gerado por quatro bombas. O sistema possui, ainda, instrumentos que realizam as medições do nível de água no vasilhame, da taxa de vapor expelida e do fluxo de água das bombas (veja Figura 1).

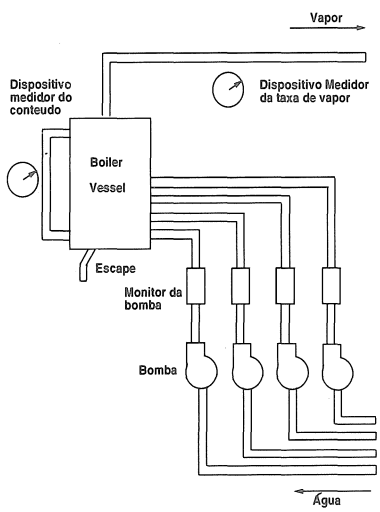


Figura 1: Boiler System

Um sistema de instrumentação (IS) realiza as medições de tais dispositivos de medida e um sistema de monitoração/controle (MCS) decide quando o sistema deve ser desligado, a partir de leituras de medidas fornecidas pelo sistema de instrumentação. Falhas podem ocorrer nos dispositivos de medida e na comunicação de mensagens entre os sistemas de instrumentação e monitoração/controle.

As mensagens que IS pode emitir a MCS são descritas completamente em [8]. Algumas delas são: *SYNC* (deverá aparecer em toda transmissão e é responsável pela sincronização entre IS e MCS), *START* (deve ser enviada por IS após a mensagem *READY* ter sido recebida, indicando que o sistema assumiu o modo **systest**), *PUMPINon/off* (informa se as bombas estão ligadas ou não), *WATERLEVEL* (informa o nível de água), entre outras. MCS também emite mensagens a IS, por exemplo, a mensagem *SYNC* (cujo parâmetro permite a verificação de mensagens atrasadas e duplicadas), *READY* (enviada quando o sistema está pronto para funcionar), *COMPTEST*, *SYSTEST*, *NORMAL*, *DEGRADED*, *EMERGENCY* e *SHUTDOWN* (informando o modo de operação

do sistema).

3 A Especificação em TLA

Utilizando uma estratégia de *comunicação de falhas*[1] e de *cálculo de nível*, as propriedades de segurança do *Boiler System* são descritas nesta seção usando a lógica TLA . Esta especificação foi desenvolvida em [1] e é utilizada aqui como ponto de partida para nossos estudos. Iniciamos com uma breve descrição de TLA.

3.1 Noções Básicas sobre TLA

A Lógica Temporal de Ações (TLA) é a combinação de duas lógicas: uma lógica de ações e uma lógica temporal simples.

Primeiramente, os conjuntos **Val** e **Var** são considerados. Assume-se que **Val** é um conjunto de valores que incluem boleanos **true** e **false**, números tais como 1, 2 e -3, sequência de caracteres como "abc" e conjuntos como o conjunto **Bool** de boleanos e **Nat** de naturais. **Var** representa o conjunto infinito de nomes de variáveis, tais como x e *boiler*.

Fórmulas atômicas em TLA são *predicados* e *ações*. Um predicado pode ser visto como uma função de *estados* em valores *booleanos*, onde um estado é uma atribuição de valores a variáveis. Uma ação é uma expressão booleana formada de variáveis, variáveis decoradas e valores. Uma ação representa uma relação entre o estado anterior (representado pelas variáveis não decoradas) e o estado posterior (representado pelas variáveis decoradas). Assim, $x' = x + 1$ é válida entre os dois estados se o valor de x no estado posterior é uma unidade maior que o valor de x no estado anterior.

A sintaxe de TLA é:

$$\mathrm{F} ::= \mathrm{P} \mid \Box A \mid \neg\ \mathrm{F} \mid F_1 \wedge F_2 \mid \Box\ \mathrm{F}$$

onde P denota predicados e A ações.

Sistemas são sempre representados em TLA por fórmulas do tipo $\mathrm{P} \wedge \Box A$, onde P representa uma condição inicial. Para expressar que uma propriedade F é satisfeita por um sistema *Sys*, escreve-se $Sys \Rightarrow F$.

3.2 Modelagem do Sistema em TLA

A especificação em TLA do *Boiler System* realizada por Bruns e Anderson em [1] usa consistência de dados para detectar falhas. O modelo do *Boiler System* em TLA captura os componentes físicos e o sistema de monitoração. O sistema de monitoração lê informações dos dispositivos e decide quando o sistema entra em **shutdown**. As bombas são ligadas pelo sistema de controle (que não faz parte do modelo). A comunicação entre os sistemas de instrumentação (responsável pelo envio de dados físicos), de monitoração e de controle, não é modelada.

As variáveis do Sistema L, L_c, *up* e *Safe* representam, respectivamente, o nível real do conteúdo do vasilhame, o valor calculado do nível, a variável que indica se o sistema está operante ou desativado e o limite de segurança do nível de água no vasilhame.

O Sistema do Boiler é descrito na forma padrão de uma especificação em TLA:

$$BSys_{TLA} \stackrel{\text{def}}{=} Init \wedge \Box Step$$

[1] Note que, *falhas* (do inglês, "*failure*"), neste contexto, dizem respeito a possíveis defeitos de dispositivos que, consequentemente, podem gerar *falhas* ("*fault*"), isto é, erros no sistema.

onde a condição inicial *Init* estabelece que, inicialmente, conhecemos o nível real do boiler que deve estar dentro dos limites de segurança e *Step* representa os componentes do Sistema.

$$Init \stackrel{\text{def}}{=} L_c = L \wedge up \wedge L \subseteq Safe$$

O modelo do *Boiler System* possui cinco componentes: o comportamento do boiler, o comportamento em caso de *shutdown*, fusão de dados, consistência de dados e determinação do nível de água.

$$Step \stackrel{\text{def}}{=} Boiler \wedge Shutdown \wedge Fusion \wedge Cons \wedge Level$$

Aqui, apenas mostraremos a modelagem, através da especificação em TLA, do componente *Shutdown*. O modelo *Shutdown* determina o valor da variável *up* que assume **true** quando o nível do boiler estiver dentro dos limites de segurança:

$$Shutdown \stackrel{\triangle}{=} up = L_c \subseteq Safe$$

Em [1], algumas propriedades matemáticas do sistema foram estabelecidas, indicando que o sistema é seguro de acordo com as hipóteses da modelagem. Dentre tais propriedades de segurança, a mais importante mostra que, enquanto a variável *up* tem o valor **true**, o nível de água do boiler permanece dentro de seu limite de segurança.
Assim, o seguinte teorema foi provado em [1]:

Teorema 1: $BSys_{TLA} \Rightarrow Safety$

onde a noção de segurança é formalizada por

$$Safety \stackrel{\text{def}}{=} \Box(up \Rightarrow L \subseteq Safe)$$

Alguns aspectos importantes não são considerados na especificação em TLA. Por exemplo, inconsistências além daquelas geradas por falhas dos dispositivos como erros de comunicação (falhas de protocolo entre subsistemas) ou em cálculos de variáveis (erros algébricos do programa) não foram consideradas. Além disso, o sistema pode assumir estados intermediários a medida que alguns dispositivos falham individualmente. Tais aspectos são considerados na especificação em Z descrita na próxima seção.

4 Transição de Estados

Analisando os pontos que não foram considerados pela especificação anterior em TLA, podemos imaginar o sistema modelado como uma sequência de transição de estados gerados por situações que comprometam a estabilidade do mesmo. Neste modelo, o tratamento dado às situações de falha varia de estado para estado. Assim, caso o sistema esteja funcionando *normalmente* e o dispositivo medidor do nível de água apresente alguma falha, o sistema assumirá o estado de *emergência*. Entretanto, caso esta falha aconteça quando o sistema estiver com outro dispositivo de medida (do vapor ou das bombas) com defeito, o funcionamento do sistema passa a ser inviável e, portanto, o sistema deve ser *desativado*. A Figura 2 mostra as possíveis transições do *Boiler System*.

Outro importante aspecto é que as transições de estado devem considerar que existem atrasos, perdas ou corrupção de mensagens no canal de comunicação entre os sistemas de instrumentação e monitoração/controle. Na verdade, o canal de comunicação entre tais sistemas apresenta um atraso inerente de transmissão referente a própria velocidade de transmissão e processamento. Desta forma, o sistema de monitoração/controle precisa estabelecer uma estimativa sobre este valor que considere a diferença entre a informação que chega e a informação real (veja Figura 2).

Quanto aos possíveis atrasos, existem três possibilidades:

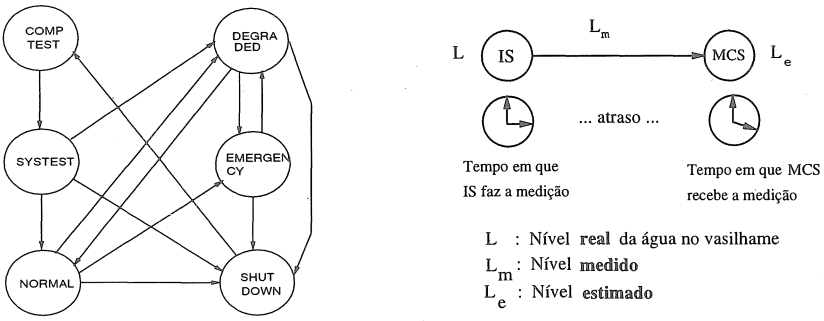


Figura 2: Transição de estados e comunicação entre subsistemas

1. *não existe falha no link*: o atraso é somente devido a velocidade de linha e processamento.

2. *mensagem chega com atraso ou é perdida*: neste caso MCS nao recebe nenhum valor e deve continuar estimando o crescimento/decrescimento através dos últimos valores recebidos.

3. *mensagem é corrompida*: neste caso MCS descarta tal mensagem e continua estimando o valor real.

4.1 A Especificação em Z do *Boiler System*

Nesta subseção, apenas alguns trechos da especificação em Z são apresentados, com o objetivo de fornecer uma idéia geral de como os aspectos dicutidos acima foram descritos. Para uma verificação completa desta especificação veja [3].

Z é uma notação baseada em lógica de predicados e teoria de conjuntos para modelagem rigorosa e construtiva de sistemas. Uma especificação em Z é formada por vários tipos de declarações para introduzir nomes, expressões que denotam termos, predicados que expressam propriedades dos valores dos nomes declarados e esquemas que impõem uma estrutura sobre as declarações e predicados. Assumimos que o leitor seja familiarizado com esta notação, descrita detalhadamente em [10].

A especificação em Z descreve o comportamento do *Boiler System* baseada na transição de seis estados representados pelo tipo enumerado *STATE*.

Por convenção, neste trabalho, um esquema é referenciado em letras estilo *itálico* e inicia-se com letra maiúscula. As variáveis são escritas em letras minúsculas salvo aquelas cujos nomes são formados por uma única letra. Para evitar dúvidas quando se fala de uma transição de estados (representada por um esquema) e um estado específico, este último será escrito em **negrito**.

As constantes *lim-sup* e *lim-inf* representam os limites físicos do nível de água no vasilhame que não poderá ultrapassar seu limite superior nem obter um nível menor que seu limite inferior. S_p e P_p representam os limites físicos da taxa de vapor máxima e fluxo total de água bombeada, respectivamente. A constante K é o fluxo de água por bomba e np, o número total de bombas no sistema (ligadas ou não). As constantes v_l, v_s e v_p são introduzidas para representar as taxas de variação mínimas das medidas do nível, vapor e bomba, respectivamente, referentes a atrasos inerentes ao canal de comunicação. Também são introduzidas as constantes i_n, i_d, i_e e i_s, que indicam os índices de incerteza de medidas provenientes dos estados **normal**, **degraded**, **emergency** e **shutdown**, respectivamente. A contante *period*, não considerada na especificação anterior, indica o intervalo de

tempo referente a cada transmissão (igual a 5 segundos, de acordo com a especificação informal em [8]).

Algumas das mensagens trocadas pelo sistema de Intrumentação e Monitoração/Controle são modeladas pelo tipo enumerado *MESSAGE*. Outras mensagens como *WATERLEVEL*, *STMRATE* e *WATFLOW*, emitidas pelo sistema de intrumentação, são representadas pelas variáveis de entrada $L_m?$, $S_m?$ e $P_m?$, respectivamente. As mensagens *COMPTEST*, *SYSTEST*, *NORMALOPS*, *DEGRADED*, *EMERGENCY* e *SHUTDOWN*, emitidas pelo sistema de monitoração/controle e que informam o estado do *Boiler System*, foram omitidas deste modelo, pois o estado do sistema é representado pela variável *state* que assume um dos valores de *STATE*.

$STATE ::= Comptest \mid Systest \mid Normal \mid Degraded \mid Emergency \mid Shutdown$
$MESSAGE ::= Ready \mid Watlevok \mid Stmratok \mid Pumpok \mid Watlevus \mid Stmratus \mid Pumpus$
$SITUATION := OPEN \mid CLOSED \qquad INDICATOR := ON \mid OFF$

$| \; Safe : \; (\mathbb{R} \times \mathbb{R}) \quad | \; K : \; \mathbb{R} \quad | \; S_p, P_p : \; \mathbb{R} \quad | \; v_l, v_s, v_p : \; \mathbb{R}$

$$\begin{array}{|l} i_n, i_d, i_e, i_s : \; \mathbb{R} \\ \hline i_n < i_d < i_e < i_s \\ i_n > 0 \end{array} \qquad \begin{array}{|l} np : \; \mathbb{N}^+ \\ \hline np = 4 \end{array} \qquad \begin{array}{|l} period : \; \mathbb{N}^+ \\ \hline period = 5 \end{array}$$

O estado do *Boiler System* é representado pelo esquema *BSys-Real* onde L, S, e P são respectivamente os valores reais do nível de água no vasilhame, da taxa de vapor e do fluxo de água bombeada. Assume-se que estes valores são constantemente alterados durante o funcionamento do sistema, embora as operações responsáveis por estas alterações não estejam representadas na especificação em Z.

L_c, S_c e P_c representam os valores do nível de água, taxa de vapor e fluxo de bombas calculados pelo sistema de instrumentação. O cálculo de medidas é considerado correto e por isso é definido em termos de valores reais do *Boiler System*. As variáveis L_e, S_e e P_e correspondem aos respectivos valores estimados calculados pelo sistema de monitoração/controle. L_{old}, S_{old} e P_{old} representam os últimos valores medidos do nível, fluxo e vapor, respectivamente, uma que vez pode haver perda de mensagens. O componente de estado *syncron* funciona como um contador de mensagens e através deste componente é estabelecida a sincronização entre os processos comunicantes.

Os componentes de estado *pumps*, *valve* e *up* indicam se as bombas estão ligadas, o escape está aberto e se o sistema está em funcionamento, respectivamente.

BSys-Real

$$\begin{array}{|l} L, S, P, L_c, S_c, P_c : \mathbb{R} \\ L_{old}, S_{old}, P_{old}, L_e, S_e, P_e : \mathbb{R} \\ pumps : Seq \; INDICATOR \\ syncron : \mathbb{N}^+ \\ valve : SITUATION \\ up : INDICATOR \\ state : STATE \\ \hline \sharp(pumps) = np \wedge (S_c \leq S_p) \wedge (P \leq P_p) \wedge (S_c \leq S_p) \wedge (L_c = L + (P - S)) \wedge (P_c = K * np) \end{array}$$

A função *FE* é definida para auxiliar no cálculo do intervalo estimado de uma medida recebida por MCS que sempre utiliza uma taxa de variação proveniente de atraso na comunicação. *plt* calcula o período decorrente desde a última transmissão.

$$\begin{array}{|l} FE : \; (\mathbb{R} \times \mathbb{R}) \rightarrow (\mathbb{R} \times \mathbb{R}) \\ \hline FE = \lambda \, m : \mathbb{R}, \; v : \mathbb{R}.(m - v, m + v) \end{array} \qquad \begin{array}{|l} plt : \; \mathbb{N}^+ \rightarrow \mathbb{N}^+ \\ \hline plt = \lambda \, s : \mathbb{N}^+.(s - syncron * period) \end{array}$$

$$\frac{- \subseteq_I - : \quad (\mathbb{R} \times \mathbb{R}) \leftrightarrow (\mathbb{R} \times \mathbb{R})}{(a, b) \subseteq_I (c, d) = (a < b) \wedge (a \geq c) \wedge (b \leq d)}$$

O esquema *Estimate-Level* especifica o controle do cálculo estimado do nível da água baseado nas informações de medidas de dispositivos, taxas de variação e verificações sobre as transmissões. Considerando tais transmissões, quatro casos são possíveis:

1. a mensagem esperada em um intervalo de transmissão é recebida ($syncron = sync?$ e $transmit? =$ **true**) com atrasos inerentes a velocidade de linha e processamento e, portanto, as taxas de variação v_l, v_s e v_p e o estado em que o sistema se encontra são considerados no cálculo do valor estimado;

2. no lugar da mensagem esperada, no mesmo período de transmissão é recebida uma mensagem antiga ($sync? < syncron$) que é descartada e, portanto, o cálculo estimado é baseado em vários aspectos: última medida recebida, o tempo de espera equivalente a um período de transmissão (considerando que a mensagem anterior foi devidamente recebida), atraso decorrente do canal de comunicação e o estado atual do sistema;

3. uma mensagem posterior a mensagem esperada é recebida ($sync > syncron$) e, portanto, além do atraso decorrente do canal e o estado atual do sistema, considera-se o tempo de espera no cálculo estimado da medida;

4. nenhuma mensagem chega e, portanto, o cálculo estimado é idêntico ao item 2.

Estimate-Level

$$L_m?, i! : \mathbb{R}$$
$$sync? : \mathbb{N}^*$$
$$transmit? : \mathbb{B}$$
$$\begin{array}{l} transmit? \Rightarrow \\ ((syncron = sync? \wedge L_e = FE(L_m?, v_l + i!) \wedge L'_{old} = L_e \wedge syncron' = syncron + 1) \vee \\ (syncron < sync? \wedge L_e = FE(L_{old}, plt(sync?) * (v_l + i!)) \wedge syncron' = sync? + 1) \vee \\ (syncron > sync? \wedge L_e = FE(L_{old}, period * (v_l + i!)) \wedge syncron' = syncron)) \\ \neg transmit? \Rightarrow (L_e = FE(L_{old}, period * (v_l + i!)) \wedge syncron' = syncron) \end{array}$$

O esquema *RLimit-Ok* verifica se os limites para o nível de água estão sendo respeitados pelo nível estimado. O esquema *BSys-Real-Off* representa o estado em que o sistema não está em funcionamento, ou seja, sistema desligado, bombas desligadas e o escape aberto.

RLimit-Ok

$$L_c : \mathbb{R}$$
$$L_c \subseteq_I Safe$$

BSys-Real-Off

$$\Delta Bsys$$
$$ran\ pumps' = \{OFF\} \wedge valve' = OPEN \wedge up' = OFF$$

O estado inicial do sistema é representado pelo esquema *Comptest*, onde o sistema está desligado e os valores zerados.

Comptest

$$Bsys\text{-}Off$$
$$msg! : MESSAGE$$
$$\begin{array}{l} L'_c = 0 \wedge L'_e = 0 \wedge L'_{old} = 0 \wedge S'_c = 0 \wedge S'_e = 0 \wedge S'_{old} = 0 \wedge P'_c = 0 \wedge P'_e = 0 \wedge P'_{old} = 0 \wedge \\ syncron = 0 \wedge state' = Comptest \wedge msg! = Ready \end{array}$$

Para ilustrar como as transições da Figura 2 são especificadas, mostramos a seguir a modelagem da transição do estado **normal** para o estado **shutdown**, caso em que os limites físicos do nível de água são ultrapassados.

Normal-To-Shutdown

$$\begin{array}{|l} \textit{BSys-Real-Off} \\ \textit{Estimate-Level} \\ \neg \textit{RLimit-Ok} \\ \hline i! = i_s \wedge \textit{state} = \textit{Normal} \wedge \textit{state}' = \textit{Shutdown} \end{array}$$

Em [3], também foram provadas algumas propriedades importantes sobre esta especificação como determinismo e ausência de *deadlock* sobre as transições de estado. Entretanto, é preciso mostrar que a especificação descrita em TLA e a especificação em Z não são *conflitantes*, ou seja, que não existe contradição lógica entre as duas descrições com relação às provas de consistência realizadas em cada uma delas separadamente. Isto será desenvolvido na próxima seção.

5 Integração entre as Especificações do *Boiler System*

Para se provar propriedades de ou estabelecer relações entre as especificações envolvendo Z e TLA é necessário definir uma relação matemática rigorosa entre os formalismos.

Um possível enfoque é fazer uma tradução da especificação em Z para uma especificação em TLA que possua o "mesmo significado". O problema deste enfoque é definir precisamente o que quer dizer "mesmo significado" quando considera-se formalismos diferentes. Uma solução seria estabelecer uma relação matemática entre as semânticas de Z e TLA, que permitisse mostrar equivalência de significados das especificações nestes formalismos. Entretanto, este problema, embora relevante, é extremamente complexo e fora do escopo deste trabalho.

O que se propõe aqui é produzir uma tradução sistemática de estruturas da linguagem Z para estruturas de TLA. A corretude semântica desta tradução é realizada intuitivamente com base na preservação de significados entre os dois formalismos, de forma semelhante ao enfoque utilizado para a tradução da linguagem MooZ para Eiffel em [7].

5.1 Tradução de Z para TLA

Para realizar a tradução de Z para TLA, uma função de tradução sintática é definida, denominada por τ, que leva um subconjunto de frases Z em TLA. Para uma melhor visualização de como uma especificação em Z é traduzida para TLA, a função τ é definida em duas etapas: tradução de tipos de dados e tradução de estruturas sintáticas.

A tradução dos tipos de Z para TLA é realizada considerando cada tipo em Z um subconjunto do conjunto **Val** de valores.

Uma especificação em Z é composta por uma seqüência linear de parágrafos que podem ser um dos seguintes objetos sintáticos: esquema, descrição axiomática, restrição, abreviação, tipo básico e tipo algébrico. Desta forma, uma especificação em Z é descrita como segue:

Especificação	::=	Parágrafo ... Parágrafo	
Parágrafo	::=	Esquema	Esquema
	|	Nome-Esquema $\triangleq$ Esquema-Exp	Esquema horizontal
	|	Descr-Axiom	Descr. axiomática

| Predicado	Restrição
| Ident == Expressão	Abreviação
| [Ident,...,Ident]	Tipo básico
| Ident ::= Ident |...| Ident	Tipo algébrico

Em [3], a função de tradução τ é definida para cada um dos parágrafos a partir de uma sintaxe simplificada, contendo unicamente as estruturas de Z consideradas no trabalho citado. Mostramos a seguir a definição de τ para algumas destas estruturas sintáticas.
Por exemplo, a função de tradução τ para um esquema é definida por:

τ *(Esquema)* $= Nome\text{-}Esquema \triangleq \tau$ *(Esquema-Def)*
τ *(Esquema-Def)* $= \tau$*(Declaração)* $\wedge \tau$*(Predicado)*

Na tradução de declarações e predicados, considera-se que os símbolos lógicos e relacionais de Z e TLA tem o mesmo nome. Assim, um predicado em Z do tipo $A \wedge B$ seria traduzido para $\tau(A) \wedge \tau(B)$ onde o símbolo $\wedge$ da tradução é a conjunção em TLA. Desta forma, temos por exemplo:

τ *(¬ Predicado)* $= \neg \tau$ *(Predicado)*
τ *(Predicado* $\wedge$ *Predicado)* $= \tau$ *(Predicado)* $\wedge \tau$ *(Predicado)*
...

Seguindo o mesmo raciocínio, a função de tradução τ é definida para as outras estruturas de Z, bem como cálculo de esquemas.

5.2 Consistência entre as Especificações do *Boiler System*

Considerando a especificação em TLA do *Boiler System*, o Teorema 1 definido na Seção 3.2 mostra que o sistema satisfaz uma importante propriedade de segurança referente ao nível de água do *Boiler System*:

$$BSys_{TLA} \Rightarrow Safety$$

Tal propriedade também deve ser preservada pela especificação em Z, quando as duas especificações do *Boiler System* são utilizadas conjuntamente. Para isso, a especificação em Z é traduzida para TLA utilizando a função de tradução τ comentada na seção anterior. Apenas parte da tradução é apresentada a seguir.

Desta forma, o modelo descrito em Z das transições de estado considerando falhas na comunicação é representado em TLA por

$$BSys_{\text{Z-TLA}}\text{-}Real \triangleq Init_{\text{Z-TLA}}\text{-}Real \wedge \Box Predicates_{\text{Z-TLA}}\text{-}Real$$

onde, $Init_{\text{Z-TLA}}$-*Real* representa as condições iniciais do sistema (em Z representadas pelo esquema que especifica o estado inicial *Comptest*) e as traduções de todas as partes de declaração (de esquemas e descrições axiomáticas), $Predicates_{\text{Z-TLA}}$-*Real* representa todos os predicados dos esquemas que modelam as transições de estado da Figura 2, incluindo o predicado invariante, definido no esquema que especifica o estado do sistema, e a parte de predicados definidas nas descrições axiomáticas. Para facilitar a leitura das fórmulas, $Init_{\text{Z-TLA}}$-*Real* e $Predicates_{\text{Z-TLA}}$-*Real* são subdivididas em fórmulas menores.
A fórmula $Predicates_{\text{Z-TLA}}$-*Real* é definida como segue

$Predicates_{\text{Z-TLA}}\text{-}Real \stackrel{\text{def}}{=} (np = 4) \wedge (period = 5) \wedge ... \wedge Inv_{Real} \wedge Transitions_{Real}$

$Transitions_{Real} \stackrel{\text{def}}{=} Comptest\text{-}To\text{-}Systest \vee Systest\text{-}To\text{-}Normal \vee ... \vee Shutdown\text{-}To\text{-}Systest$

onde, por exemplo, considerando o esquema *RLimit-Ok* definido por

$RLimit\text{-}Ok \stackrel{\text{def}}{=} L_c \subseteq Safe$

tem-se a transição *Normal-To-Shutdown* traduzida para a fórmula a seguir

$$\begin{array}{lll} Normal\text{-}To\text{-}Shutdown & \stackrel{\text{def}}{=} & \neg\ RLimit\text{-}Ok \\ & \wedge & BSys\text{-}Off \\ & \wedge & i! = i_s \\ & \wedge & state = Normal \\ & \wedge & state' = Shutdown \end{array}$$

Uma vez traduzida a descrição do *Boiler System BSys-Real* para a fórmula em TLA $BSys_{\text{Z-TLA}}$-*Real* é necessário estabelecer que $BSys_{\text{Z-TLA}}$-*Real* satisfaz as condições de segurança da especificação em TLA. Desta forma, seria preciso provar que:

$$BSys_{\text{Z-TLA}}\text{-}Real \Rightarrow Safety \qquad (1)$$

No entanto, isto não pode ser realizado diretamente pois as variáveis em $BSys_{\text{Z-TLA}}$-*Real* e *Safety* podem não ser as mesmas, terem tipos diferentes ou simplesmente denotar objetos diferentes apesar de terem o mesmo nome. Em particular, a variável *up* existe nas duas fórmulas mas com tipos diferentes:

- *up* em $BSys_{\text{Z-TLA}}$-*Real*, obtido através da função de tradução τ a partir da variável da especificação em Z, assume os valores em $\{ON, OFF\}$,
- *up* em *Safety*, é o *up* da especificação em TLA, que é uma variável boleana, ou seja, assume valores em $\{\mathbf{true}, \mathbf{false}\}$.

É claro que existe uma bijeção entre os dois conjuntos de valores para *up*. Desta forma, pode-se facilmente obter uma fórmula em TLA equivalente a (1), que define o teorema a seguir:

Teorema 2: $BSys_{\text{Z-TLA}}\text{-}Real \Rightarrow \Box(up = ON \Rightarrow Lim\text{-}Ok)$

Para facilitar a demonstração da prova do Teorema 2, o seguinte lema é definido

Lema 1:

$Transitions \Rightarrow (up = ON \Rightarrow Limit\text{-}Ok)$

Em [3], o Lema 1 é provado facilmente através do método de tableau [9]. Assim, utilizando o Lema 1 e a regra de inferência TLA STL4 descrita como segue

STL4. $\frac{F \Rightarrow G}{\Box F \Rightarrow \Box G}$

a prova do Teorema 2 é demonstrada a seguir:

Prova do Teorema 2:

$Transitions$	$\Rightarrow$	$(up = ON \Rightarrow Limit\text{-}Ok)$	pelo Lema 1
$\Box Transitions$	$\Rightarrow$	$\Box(up \Rightarrow Limit\text{-}Ok)$	pela regra STL4
$\Box Predicates \wedge Init_Z$	$\Rightarrow$	$\Box(up \Rightarrow Limit\text{-}Ok)$	lógica proposicional
$BSys_Z$	$\Rightarrow$	$\Box(up \Rightarrow Limit\text{-}Ok)$	pela definição de *BSys-Real*

□

Desta forma, podemos afirmar que a especificação em Z garante propriedades de segurança da especificação em TLA, e portanto, quando utilizadas em conjunto, as especificações não são contraditórias, como queríamos demonstrar.

6 Considerações Finais

O objetivo básico deste artigo é mostrar que a utilização de múltiplos formalismos permite especificar requisitos iniciais de sistemas complexos.

A partir de uma descrição informal de um sistema de caldeiras (*Boiler System*), descrevemos dois modelos distintos nos formalismos TLA e Z. Mostramos que a especificação em TLA, tomada isoladamente, deixa de considerar aspectos importantes do sistema como transição de estados e comunicação entre subsistemas controladores. Com a especificação em Z, aumentamos o conhecimento sobre o sistema e, consequentemente, a precisão dos requisitos nesta fase de desenvolvimento. Desta forma, desenvolvemos uma especificação do *Boiler System* mais completa e, com as provas de consistência entre as duas especificações, mais confiável.

Portanto, comprovamos, teoricamente (visto que não chegamos a uma implementação do *Boiler System*), a viabilidade e adequação da utilização de múltiplos formalismos na análise de requisitos de sistemas.

Alguns aspectos como o custo de desenvolvimento formal e o uso de ferramentas de apoio não são discutidos aqui, embora reconhecemos a importância de tais considerações.

Referências

[1] Glenn Bruns and Stuart Anderson. Using Data Consistency Assumptions to Show System Safety, 1993. Laboratory for Foundations of Computer Science, Edinburgh.

[2] R. de Lemos, A. Saeed, and T. Anderson. A Train as a Case Study for the Requirements Analysis of Safety-Critical Systems. *Computer Journal*, 35(1):30–39, 1992.

[3] Simone C. dos Santos. Especificação de sistemas críticos: um enfoque baseado na utilização consistente de múltiplos métodos formais. Master's thesis, Universidade Federal de Pernambuco, November 1995.

[4] Simone C. dos Santos e Fabio Q. B. da Silva. Um Enfoque Multiformalismos para Especificação de Sistemas de Segurança Crítica. *IX Simpósio Brasileiro de Engenharia de Software*, 1995.

[5] Ismar N. Kaufman. A Aplicação de Especificações Formais no Projeto Lógico de Software. Master's thesis, Universidade Federal de Pernambuco, Recife - Brasil, 1992. Departamento de Informática.

[6] Leslie Lamport. The Temporal Logic of Actions. Technical report, Digital Systems Research Center, 1991.

[7] Virgínia Adélia de Oliveira Cordeiro. De MooZ para Eiffel - Uma Abordagem Rigorosa para o Desenvolvimento de Sistemas. Master's thesis, Universidade Federal de Pernambuco, Recife, Pernambuco, Brasil, 1994.

[8] Institute For Risk Research. Specification for a software program for a boiler water content monitor and control system. Waterloo, Canadá, 1992.

[9] R. M. Smullyan. *First-Order Logic.* Springer-Verlag, Berlim, 1968.

[10] J. M. Spivey. *The Z Notation - A Reference Manual.* Prentice Hall, 1992.

Experiência do GMC no desenvolvimento de software instrucional

Tiarajú Asmuz Diverio+
Instituto de Informática e CPGCC da UFRGS
diverio@inf.ufrgs.br

Beatriz Regina Tavares Franciosi+
Instituto de Informática da PUCRS e UFRGS
bea@inf.ufrgs.br

ABSTRACT

This paper presents a software development background owner to Computing Mathematics Group (GMC) at Instituto de Informática - UFRGS (Brazil) and its use on a software system design called LEPMAC. This system is an instructional software, which takes account teach and research experiences. The aim of this software is support the teach process in numerical analysis and related areas. In the same way, the software can be used to assist some research activities. First of all, is present the GMC's software development background following by a full LEPMAC description. Finally, some considerations about problems on the LEPMAC development and maintenance are done.

INTRODUÇÃO

Há algum tempo, por necessidade de pessoas que utilizam computações numéricas, começou a surgir *software* com o intuito básico de solucionar problemas envolvendo matemática numérica. A princípio foram desenvolvidos pequenos programas para solucionar problemas específicos. Com o passar do tempo, estes programas foram agrupados de acordo com áreas específicas, constituindo pacotes de *software*, cujo aperfeiçoamento gerou bibliotecas e sistemas de *software*.

O uso de bibliotecas, geralmente pressupõe que o usuário tenha um bom conhecimento do problema a resolver e de programação, a fim de que possa acessar e incluir a biblioteca numérica. Isto restringe bastante o número de usuários e remete ao desenvolvimento de *software* para processamento numérico que libere do conhecimento de linguagens de programação.

O desenvolvimento deste tipo de *software* é especialmente importante para disponibilizar ferramentas de apoio ao ensino de matemática computacional, onde a visualização do processamento numérico é fundamental para compreensão de conceitos.

O sistema **LEPMAC** (Laboratório de Ensino e Pesquisa em Matemática Aplicada e Computacional) é um *software* para processamento numérico, o qual disponibiliza uma interface amigável com o usuário e que implementa um conjunto básico de métodos aproximativos. Este

+ suported by CNPq

sistema também disponibiliza a visualização gráfica de funções, a fim de que o usuário possa verificar graficamente zeros polinomiais calculados numericamente.

A equipe responsável pelo desenvolvimento do projeto constituiu o Grupo de Pesquisa em Matemática da Computação da Universidade Federal do Rio Grande do Sul (GMC-UFRGS), vinculado ao Instituto de Informática e ao Curso de Pós-Graduação em Ciência da Computação.

O LEPMAC é um sistema instrucional constituído por um conjunto de rotinas implementadas modularmente. A interface usuário-sistema é orientada a cardápios do tipo *pull-down* os quais disponibilizam diferentes níveis de atividades.

O cardápio principal é constituído pela identificação das classes de métodos implementados, nas quais estão incluídos diversos métodos aproximativos. Como o sistema é de propósito educacional, incluiu-se métodos de excelente qualidade, sob o ponto de vista de eficiência e outros de qualidade questionável. O objetivo foi proporcionar situações de análise e de reflexão sobre o uso de métodos aproximativos, no que se refere a tempo de processamento, número de operações realizadas e exatidão de resultados fornecidos.

O software é auto-explicativo, através de telas que informam sobre o conteúdo (métodos implementados) e recomendação de uso de métodos numéricos.

A organização do sistema está baseada em módulos, os quais são independentes entre si e integrados por um gerenciador. O gerenciador é responsável pela interface usuário-sistema. Ele é composto por um conjunto de telas explicativas do uso do sistema; informativos do sistema (contendo: objetivos, filosofia e o endereço para contatos). Informações do conteúdos dos módulos podem ser obtidas via tela de ajuda existente no sistema.

A interface usuário-sistema dispõe de atributos como simplicidade, isomorfismo interno (uso de mesmo comando sempre que a ação é a mesma), isomorfismo externo (entre interfaces), formato estruturado de tela, uso de modelos conceituais familiares aos usuários, possibilidade de desfazer ações. Através do cardápio principal estão disponíveis os seguintes módulos:

- **POL** - O módulo de Equações Polinomiais possibilita a resolução de polinômios de grau n (onde n é menor ou igual a vinte). Nesse módulo estão implementados métodos de determinação e localização das raízes reais e complexas, bem como a possibilidade de visualização gráfica. Também é possível ativar técnicas para deflação de raízes de polinômios;

- **SIGMA** - O módulo possibilita análise de funções algébricas e transcendentes; pode-se traçar esboços gráficos de funções e suas derivadas, calcular os zeros de funções por diferentes métodos.

- **SSE** - O módulo de solução de Sistemas de Equações trata da solução de sistemas de equações lineares de variáveis reais e complexas. O pacote resolve sistemas de ordem n(onde n é menor ou igual a vinte), ou sistemas complexos de ordem n/2 (ou seja menor ou igual a dez).

Estão disponíveis as funções de cálculo de condicionamento, determinante e o cálculo da matriz inversa. Foram implementados métodos diretos, compactos e iterativos. Nesse módulo é possível efetuar refinamentos da solução.

- SONDA - O módulo trata de aproximações de funções por ajuste de mínimos quadrados e interpolação polinomial. Pode-se ter até n pontos de entrada para se aproximar a funções (para a versão atual, n é igual a vinte). Estes pontos podem ser plotados ou ajustados a funções do tipo: reta, parábola, exponencial e potência. Pode-se verificar o melhor ajuste pelo esboço gráfico ou pela soma dos mínimos quadrados; pode-se, ainda, dado um valor X, achar seu Y correspondente ou vice-versa. Quanto aos métodos de interpolação, estão disponíveis os métodos de Lagrange e Newton por diferenças divididas e finitas.

- SPL - O módulo trata do módulo de *Splines* cúbicos, pode ser utilizado para interpolar funções por *Splines*.

- STAT - Entre as funções que estão disponíveis no módulo Estatístico, destaca-se as da Estatística Descritiva como medidas de tendência central (média aritmética simples, mediana, quartis, moda, média geométrica e média harmônica), medidas de dispersão (amplitude total, variância absoluta, propriedades da variância e desvio-padrão), medidas relativas de variabilidade (variância relativa coeficiente da variação e variável padronizada). No que se refere à amostragem há geração de números aleatórios, estimadores (média, variância de estimadores para população finita ou infinita, erro-padrão do estimador, proporção, variância do estimador para populações finitas ou infinitas e erro-padrão), intervalos de confiança para proporção, testes de hipóteses estatísticas (sobre parâmetros de distribuições normais, para uma média ou comparação entre duas médias e teste para proporções).

O capítulo de integração numérica, devido a necessidade de se introduzir funções algébricas e transcendentes, ficou vinculada ao módulo SIGMA, que possui um interpretador de funções. Os métodos de integração disponível são método dos Trapézios, método de Simpson, de Romberg e o da quadratura gaussiana.

O sistema foi desenvolvido em turbo Pascal (versão 5.5), uma vez que ele dispõe de rotinas gráficas capaz de suportar os aspectos gráficos do sistema e da interface. Estão disponíveis três versões do sistema: uma para disco rígido; uma para dois *drives* e uma para rede local.

O LEPMAC dispõe de um conjunto de manuais e apostilas didáticas de apoio. Entre este material que compõe a documentação destaca-se a Cartela de Rápida Utilização do Sistema, Manual de Utilização, Caderno de Exercícios e as Apostilas didáticas para os módulos de Polinômios e Sistemas de Equações.

RELATO DA EXPERIÊNCIA DO GMC-UFRGS

A primeira experiência em software matemático na informática da UFRGS foi em 1981, em um projeto de documentação da biblioteca de rotinas matemáticas **NUMERALS** existente no computador central B6700 (Bourroughs). O NUMERALS é uma biblioteca de rotinas numéricas em Algol e Fortran para álgebra linear e solução de sistemas de equações lineares.

Em 1982 começou-se a desenvolver em Pascal um pacote de rotinas matemáticas para a resolução de equações não lineares e sistemas de equações lineares. O ambiente de desenvolvimento deste pacote foi um minicomputador LABO8037 (alemão - Nixdorf). O compilador Pascal utilizado foi produto de uma dissertação de mestrado de G. Medeiros, uma versão *standard* do Pascal bastante limitada em termos de recursos de entrada e saída de dados, pois só lia e imprimia caracteres. A idéia de interface era igual a cardápios de opções onde elas são acessadas por um número ou uma letra.

Em 1985, com a chegada dos microcomputadores de 8 *bits* e posteriormente, os de 16 *bits*, iniciou-se o desenvolvimento de estudos sobre software numérico instrucional e o desenvolvimento de um pacote instrucional em BASIC. Observa-se que nesta época o BASIC para PC era a única linguagem com recursos gráficos disponível no Brasil. O pacote foi denominado **SINAI-16** (Software Interativo Numérico Aplicativo e Instrucional em micros de 16 *bits*) e dispõe de rotinas numéricas para resolução de equações polinomiais, equações algébricas e transcendentes.

O **SINAI-16** tem um módulo gráfico para traçar esboços gráficos (e imprimi-los). Ele contém um estudo completo do capítulo de resolução de equações, incluindo localização de raízes, enumeração através da visualização gráfica e, em alguns casos, através de resultados obtidos, a determinação de valores iniciais para métodos iterativos; o cálculo de raízes pelos tradicionais e até a deflação de raízes pelo método de Briot-Ruffini e Horner. Maiores informações podem ser encontrados nos manuais ou nos artigos indicados na referência. A interface do **SINAI-16** é baseada em cardápios onde as ações são selecionadas por letras ou números. As mensagens explicativas são apresentadas apenas no início do software e não dispõe de sistema de ajuda. A consistência na ativação das funções é fraca, pois a cada tela a função de retorno é ativada por um número ou uma letra diferente. Existe consistência a erros da entrada de dados, pois uma vez digitada uma informação inválida o software alertava e solicitava que fosse re-introduzido o dado ou parâmetro.

Este software foi a primeira experiência efetiva de desenvolvimento de software instrucional do grupo GMC. O software foi utilizado no apoio ao ensino de disciplinas numéricas do terceiro grau, em cursos de matemática, engenharias e computação de diversas instituições de ensino superior. A partir dele iniciou-se várias outras experiências, inclusive um projeto de desenvolvimento de um software instrucional em computador de grande porte.

O **SISSEL** - *Software* Instrucional para Solução de Sistemas de Equações Lineares, foi outro pacote desenvolvido com a finalidade instrucional, o qual possibilita solução de sistemas de equações, tanto reais quanto complexas. Ele tem um módulo de leitura, verificação, e correção de

dados (coeficientes das matrizes); pode resolver o sistema por vários métodos, como por exemplo: Cramer, Eliminação de Gauss, Gauss-Jordan, por Métodos compactos ou iterativos. Ele permite a verificação da estabilidade do sistema através de uma pequena modificação nos valores do termo independente, e analisando a conseqüência no resultado ou pelo cálculo da medida de condicionamento do sistema. A interface é semelhante ao do SINAI-16, pois este *software* constitui uma evolução das idéias que nortearam os sistemas anteriores.

O SIAD - Software Interativo de Ajuste de Dados, foi desenvolvido em 1987, por um grupo de alunos de informática orientados pelo Grupo de Matemática da Computação. Abrange o capítulo de ajuste de dados, sendo incorporados métodos de ajuste a funções do tipo: reta, parábola, potência e exponencial. Ele tem um módulo gráfico que possibilita a plotagem dos pontos e da curva para se observar a qualidade do ajuste. É possível também sobrepor os diferentes ajustes, a fim de determinar o melhor ajuste (esta verificação pode ser feita pela soma dos quadrados dos resíduos que é fornecida pelo programa). Existe um módulo para cálculo de estimativas de valores e regressão linear.

O SIAD foi desenvolvido em Turbo Pascal versão 3.0. O conceito de interface já estava bem mais assimilado (pois as telas seguiam um *layout)*, dispunha de um sistema de ajuda em diversos níveis contendo mensagens explicativas sobre os métodos. O projeto das mensagens do sistema teve como objetivo disponibilizar informações do manual de utilização na tela, em tempo de execução.

O projeto de interface usuário-sistema do SIAD utiliza o conhecimento adquirido, isto é, interface orientada a cardápios com escolha através de apontamento direto via teclas vetores e destaque da opção ativa em *vídeo-reverso*.. Utilizou-se ainda, o *beep* (recurso sonoro) para alertar informação inválida ou a espera de informação.

Em 1988, a experiência acumulada sobre desenvolvimento do *software* numérico já era bastante significativa e grande parte dos sistemas desenvolvidos pelo GMC-UFRGS foram cedidos a diversas universidades para apoiar o trabalho de atualização de professor e de praticas docentes.

Desta feita, a qualidade da interface usuário-sistema passou a ser uma necessidade, pois o bom uso dos sistemas dependeria da adequada compreensão de uso. O estudo tomou forma de investigação através da dissertação de mestrado, de D.L.Maçada ([MAÇ94]), cujo produto foi o desenvolvimento de um sistema para ensino de matemática que explora a linguagem visual de sistemas de computação. A idéia principal é liberar o usuário do aprendizado sobre a manipulação do sistema e priorizar aspectos relativos ao ensino-aprendizagem.

Nesse ambiente aprender a pensar é fato dominante deixando para um plano de menor importância a navegação no sistema. Desta forma, o sistema projetado libera ao máximo usuário do aprendizado de sua manipulação. A navegação é feita através de seleção em cardápios, pois estes constituem um excelente auxílio às nossas limitações de memória (lembrar é mais difícil do que reconhecer) e propiciam um menor número de erros de digitação. Os itens dos cardápios

utilizam a complementaridade da leitura da palavra e do desenho (*ícone*) e as ações que o sistema está apto a executar são especificamente gráficas.

O Pacote **SIGMA** é um exemplo onde os estudos sobre interface foram aplicados e com sucesso. Ele foi desenvolvido em Turbo Pascal (versão 4.0 no módulo gráfico), no ambiente de PC AT com vídeo colorido e co-processador aritmético. Ele abrange a parte da análise de funções matemáticas e cálculo de raízes de equações algébricas e transcendentes. O projeto de interface do SIGMA obedece, em linhas gerais as regras e convenções estipuladas para o sistema de software **LEPMAC**.

Teve-se a idéia de reunir em uma única ferramenta pacotes instrucionais que resolvem um amplo elenco de problemas numéricos. A filosofia de desenvolvimento foi a de projetar módulos instrucionais e reuni-los através de um módulo gerenciador em um único sistema. Cada módulo engloba um capítulo da matemática computacional. Este tipo de estrutura de sistema possibilita que se faça *upgrade* dos módulos individualmente, bem como a inserção de novos módulos.

Padronizações foram feitas no planejamento e desenvolvimento dos módulos, como por exemplo, o formato das telas, as mensagens de ajuda (*help*) e no modo de seleção de opção. Houve explícita preocupação com a ergonomia da interface usuário-sistema a qual é um misto de seleção em cardápio (para dados não numéricos) e entrada direta via digitação (para dados numéricos). A navegação no sistema é feita através de seleção em cardápios.

Finalmente em 1992, foi concluída a implementação do sistema LEPMAC. O sistema foi projetado para interagir com o usuário de uma forma amigável e instrucional; através de uma sobreposição de telas que contém as funções disponíveis a cada momento. A sobreposição das telas possibilitam um *overhead* do sistema. Neste sistema estão disponíveis rotinas para resolução de equações algébricas e transcendentes, sistemas de equações lineares, interpolações, ajustes de dados, integração numérica entre outros.

CONSIDERAÇÕES SOBRE USO DO LEPMAC

O sistema LEPMAC tem sido utilizado em diferentes instituições de ensino superior, a saber: Instituto de Informática da PUCRS, Laboratório de Informática da ULBRA, Institutos de Matemática e Informática da UFRGS. Também foram fornecidas versões do software para a UFSC (Florianópolis), UFC (Fortaleza), UEL (Londrina), UEM (Maringá) e PUCMG (Belo Horizonte).

Nestas instituições o sistema é utilizado como uma ferramenta de apoio para aulas práticas, onde os exercícios fornecidos devem ser resolvidos com o auxílio do sistema de software. Como o sistema possui a possibilidade de fornecer os principais valores parciais na resolução de problemas, os alunos inicialmente se utilizam desta opção, para uma melhor compreensão do problema e da sistemática de resolução e, depois, resolvem da forma mais eficiente.

Observou-se que, quando o sistema é utilizado, o rendimento dos alunos e o interesse pelos conteúdos abordados aumenta. Isto porque, o sistema homogeniza a ferramenta, o instrumento de resolução dos problemas em sala de aula, isto é, em sala de aula cada aluno tem sua calculadora com características específicas diferentes (científica, programável, com notação polonesa reversa, *pocket computers* com linguagem BASIC ou pseudo *Assembler*).

Através do sistema é possível explorar por exemplo, aspectos como instabilidade numérica. A questão da instabilidade em polinômios, pode ser ilustrada graficamente, através do gráfico do polinômio que está sendo analisado. Introduz-se a perturbação num dos coeficientes e, volta-se ao gráfico do polinômio perturbado, onde se observa a variação das raízes. Uma vez visualizado o problema, pode-se calcular pelos métodos numéricos disponíveis as raízes reais e complexas e, ainda, determinar o erro contido nas soluções.

De forma análoga, pode-se estudar a instabilidade numérica de sistemas de equações lineares com o auxílio do LEPMAC. O estudo pode abranger o conceito, ou a quantificação do problema, ou ainda, a constatação dos resultados. O conceito pode ser estudado ao variar-se os coeficientes da matriz A ou o vetor dos termos independentes. Então se calcula a variação produzida no resultado. Pode-se, com isso, comparar a taxa da perturbação introduzida e a taxa da variação produzida. Este é o princípio da análise sensitiva aplicada a sistemas de equações.

Uma das possibilidades de uso deste sistema instrucional está relatada a seguir. Como o LEPMAC calcula automaticamente a matriz inversa, determinante e os diferentes tipos de normas matriciais, pode-se verificar o condicionamento da matriz, seu relacionamento com o determinante e suas conseqüências práticas no números de algarismos significativos corretos do resultado.

Este sistema constitui uma importante ferramenta para o ensino, pois possibilita ao professor apoio na preparação de aulas, elaboração de exercícios e provas. Na verdade, o sistema de software dispõe de um Cadernos de Exercícios abrangendo os diferentes módulos do sistema, onde o professor pode selecionar o exercício e verificar se sua solução satisfaz as exigências de complexidade de resolução, dificuldade e de tempo necessário para resolução.

A representação gráfica utilizada pelo sistema evita sobrecarregar o aluno com a exibição de grande volume de dados numéricos e enfatiza a visualização de dados através de gráficos.

Percebe-se que o uso de *software* instrucionais em sala de aula, como uma ferramenta para o ensino resulta na necessidade de reformulação do plano de ensino tradicional, na forma de avaliação dos alunos e na atualização de professores. Uma das maiores resistências ao uso de ferramentas computacionais, ou mesmo de computadores no ensino vem dos professores, que não se sentem preparados suficientemente.

Para a validação do sistema de *software*, utilizou-se um conjunto de exercícios abrangendo o conteúdo dos módulos, o qual gerou um dos materiais didático do LEPMAC, o Caderno de Exercício ([DIV90a]). Ele é composto por exercícios de fixação, exemplos protótipos já mencionados na literatura, exemplos de problemas de instabilidade numérica e, exercícios para comparação do desempenho dos métodos. Este caderno visa servir como uma outra ferramenta

auxiliar para o professor utilizar na preparação de tarefas a serem realizadas pelos alunos. Também foram desenvolvidos materiais didáticos de apoio, para os módulos, os quais serviram para validar o sistema ([DIV90b, DIV90c]).

CONSIDERAÇÕES FINAIS

Dentre as inúmeras dificuldades encontradas no desenvolvimento e manutenção do LEPMAC destacaram-se duas: desenvolvimento de manuais e manutenção do software. Isto pode ser atribuído ao fato da equipe de desenvolvimento não possuir dedicação exclusiva e à complexidade associada ao desenvolvimento de software numérico.

O maior legado deste trabalho foi produzir uma ferramenta para apoio ao ensino que privilegia a visualização dos erros de programas numéricos. Desta forma, pode-se introduzir estes conteúdos de uma forma rápida e eficiente. Os estudantes podem aprofundar e expandir seus conhecimentos de tópicos de análise numérica através do uso do LEPMAC.

Nas escolas destes novos tempos, o professor deve ser encarado como um guia que conduz através das trilhas do interesse de cada aluno. Ele deve ser o profissional capaz de promover uma mudança no estado de compreensão de conceitos. O LEPMAC propõe a aprendizagem através da auto-descoberta e, além disto, ele se adequa para auxiliar no treinamento, suporte e pesquisa na área de matemática numérica.

Pensar na educação como forma de transferência de conhecimento do professor para o aluno, já não é mais possível. Os alunos devem saber não apenas os conteúdos, mas também ter o domínio das próprias formas de aprender. A educação, vista sobre este enfoque, não é simplesmente a preparação para uma carreira; ela é um empreendimento para durar a vida inteira.

BIBLIOGRAFIA

[CLA90] CLAUDIO,D.M; DIVERIO,T.A; FRANCIOSI,B.R.T. Software Instrucionais em matemática - experiência e evolução na Informática da UFRGS. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 1, 15-22, nov. 1990. Rio de Janeiro, **anais**, Rio de Janeiro, RJ, nov.19-21, 1990. pp.124-136.

[DIV86a] DIVERIO, T.A **Software Numérico aplicativo e instrucional** Porto Alegre: CPGCC da UFRGS, 1986. (dissertação de mestrado).

[DIV86b] DIVERIO, T.A **SINAI-16 - Manual Instrucional do usuário e Manual de Manutenção do Programa.** Porto Alegre: CPGCC da UFRGS, 1986.

[DIV87] DIVERIO,T.A. Resolução de equações - uma metodologia de estudo. In: Conferência Latinoamericana de Informática (CLEI PANEL'87), XIII, Bogota, nov.9-13, 1987.pp.190-203. (trabalho)

[DUA87] DUARTE,R.F et al. **SIAD - Manual do usuário**. Porto Alegre: PUCRS, 1987.

[DIV90a] DIVERIO,T.A. et al. LEPMAC - Caderno de Exercícios do Sistema de Software do Laboratório. Porto Alegre: UFRGS, 1990.

[DIV90b] DIVERIO,T.A. LEPMAC - Material Didático de Apoio - módulo de REATS. Porto Alegre: UFRGS, 1990. 62p.

[DIV90c] DIVERIO, T.A. LEPMAC - Material Didático de Apoio - módulo de SELAS. Porto Alegre: UFRGS, 1990. 61p.

[FRA88] FRANCIOSI,B.R.T **Projeto de interfaces gráficos para o ensino de deficientes auditivos**. Porto Alegre: CPGCC UFRGS, 1988. (dissertação de mestrado)

[FRA91] FRANCIOSI,B.R.T; DIVERIO,T.A; CLAUDIO,D.M Oraculus system: an education program in interactive information retrievail system. In: INTERNATIONAL CONFERENCE ON TECHNOLOGY AND EDUCATION, 8, 1991, May 3-10, **Proceedings**. The harbour Castle Westin.

[FRA92] FRANCIOSI,B.R.T; MAÇADA,D.L O ensino e o computador no ensino.In: CONGRESSO IBERO AMERICANO DE INFORMÁTICA EDUCATIVA, 1992, **anais**, Santo Domingo: 1992. p.44-53.

[MAÇ94] MAÇADA, D.L. **Métodos Numéricos Computacionais**. Porto Alegre: CPGCC da UFRGS 1994. (dissertação de Mestrado)

RECOGNIZING MATCHING AMONG TERMS: A NEURAL NETWORK APPROACH

PART A: BUILDING THE NETWORKS

Carlos Mareco *, Alberto Paccanaro * ‡

* Laboratorio de Electrónica Digital
Universidad Católica "Nuestra Señora de la Asunción"
Asunción, Paraguay
Tel: +595-21-334650
Fax: +595-21-310587
E-mail: *cmareco@ledip.py*

‡ Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Milano, Italy

ABSTRACT

This project aims at building Neural Networks that would receive as inputs (n+1) terms $\{T_1,..T_n, T\}$ and would suggest which T_is are matched by a subterm of T, with varying n. This first part of the paper, presents a particular net that can solve the problem for n=1, with a good degree of accuracy. This net was then used as a building block for larger nets that were applied to solve the problem for higher values of n.

1. INTRODUCTION TO THE PROBLEM

We assume the reader to be familiar with the basic concepts of First Order Logic (FOL) and Term Rewriting Systems (TRS). For a full coverage of the subject, see (Knuth & Bendix, 1970; Hullot, 1980; Hsiang & Rusinovitch, 1987; Dershowitz, 1989; Chang & Lee 1976).
In previous papers (Paccanaro & Willigs, 1994; Paccanaro, 1995) a particular Neural Network and a method for codifying terms and rules belonging to the complete TRS for the Group Theory were

presented; the net, given a term belonging to such theory, could suggest with a good degree of accuracy a rewrite rule belonging to the complete TRS that could reduce the term.
Notice that such net is therefore suggesting a rule of the TRS whose left hand side (lhs) is *matched* by a subterm of the term: it is actually identifying matching pairs.
The results obtained encouraged us to further investigate this area, and we focused on the Matching relation among FOL terms using Neural Networks.
Let us give a formal definition of Matching:

Matching: a term **t** is said to match a term **l** if there exists a substitution σ such that $\mathbf{l}\sigma = \mathbf{t}$.

The problem (1) we decided to investigate, can be stated as follows:

Let T(X,F) be the set of terms which can be built from a (finite) set of function symbols F (we assume constants to be 0-ary function symbols), and a (countable) set of variables X. Let $\{T_1, T_2, T_3, \ldots, T_n, T\}$ *where* T_i, $T \in T(X,F)$ *and n=1, 2, ..., be a set of terms such that there is only a term* T_k *that is matched by a subterm of T. The problem is to determine such* T_k.

We tried to find a class of neural networks that would receive as inputs (n+1) terms $\{T_1,\ldots,T_n,T\}$ and would suggest which T_k is matched by a subterm of T, with varying n.
It is worth to point out how the problem previously investigated in (Paccanaro & Willigs, 1994) turns out to be a particular case of this more general one: if we let the T_i be terms constituting the LHS of the rules of a completed TRS, and T the term to be rewritten with respect to the TRS, we get the previous problem back. Solving this more general problem would allow us to guide term rewriting with respect to any TRS.

2. EXPERIMENTAL SETUP

2.1 Databases of terms.

We decided to limit the problem to terms of length 7 or less, generated from a set of 3 function symbols: f, i and e with arity 2, 1 and 0 respectively.
Seven different databases of terms were built to be used in the experiments. Four of them were generated using a number of distinct variables varying between 1 and 4 (using the functions symbols described above and limiting the length of the terms to 7, a term may have at the most 4

distinct variables). In the following these databases will be indicated as DBi, where i = 0, ..., 4 stands for the number of distinct variables which were used for building the terms.

Then 3 more databases were generated from the ones containing 2, 3, and 4 variables by eliminating terms that could be made equal through variable renaming. This allowed to generate databases of terms which were smaller than the original ones leaving unchanged the total number of possible different variables present in each term.

In the following these databases will be indicated as DBRi, where i = 2, 3, 4 stands for the number of distinct variables which were used for building the terms before renaming.

The following table presents some relevant features of the composition of the databases which will be used in the subsequent discussions.

Database	# of terms	avg. # of var for term	avg. length of term
DB1	570	1.4228	6.4842
DB2	1875	2.0171	6.5968
DBR2	982	1.9257	6.5794
DB3	4556	2.3593	6.6620
DBR3	1074	2.0456	6.6024
DB4	9285	2.5878	6.7060
DBR4	1079	2.0547	6.6043

Table 1

It is worth to notice that DB4 can be considered as the "most general" database, being the largest and the one containing all the others.

2.2 Term Codification

Terms were codified as vectors of integer numbers of length 7 using the method that was found to yield best results in (Paccanaro & Willigs, 1994).

A positive integer number was assigned to each symbol and each element of the codified vector contained the number associated to the symbol in that position of the term. The vector was filled up starting from its last position, while a value of 0 was assigned to the elements not used for codifying the term.

In the following we shall present results of experiments which make use of a codification for terms which assigns the values of 500, 700 and 900 to the symbols "f", "i", and "e" respectively, while all the variables are assigned to 0. According to this codification, the vector corresponding to the term "f(i(x), f(e))" would turn out to be:

[0, 0, 500, 700, 0, 500, 900].

2.3 Networks Parameters

The learning algorithm used was the Normalized Cumulative Delta Rule. The processing elements used the hyperbolic tangent as transfer function. No noise was added to the inputs.

All nets were trained with a number of iterations between 200,000 and 300,000. The initial values used for the learning coefficient and momentum were 0.7 and 0.6 respectively. These values were gradually reduced as the learning process progressed.

Also, it is important to notice that the processing elements of the networks did not actually receive as inputs numbers such as 500 or 700, as they would soon saturate the neurons. To prevent this, the values used in the codification were scaled between -1 and 1, using linear scaling.

3. RECOGNIZING MATCHING AMONG PAIRS OF TERMS

We began trying to build a net, such that it could solve the problem for n=1. The net would receive as input 2 terms, T_1 and T, and should decide whether T_1 is matched by a subterm of T.

The net will have therefore 14 input elements (2 terms of length 7) and 1 output element.

For each database of terms a set of pairs of terms was built, according to the following procedure: for each term T in the database, two terms T_A, T_B were randomly chosen (from the same database), such that T_A was matched by a subterm of T while T_B was not; these terms were used to build 2 pairs [(T_A; T), 1], and [(T_B; T), 0]. From each set thus obtained, 75% of the pairs were randomly chosen to be used for training, while the remaining 25% were kept for testing purposes.

Various net architectures were tried. During testing the output of the net was considered as 1 when above 0.5, and 0 otherwise. The network architecture that gave best results is shown in Fig 1.

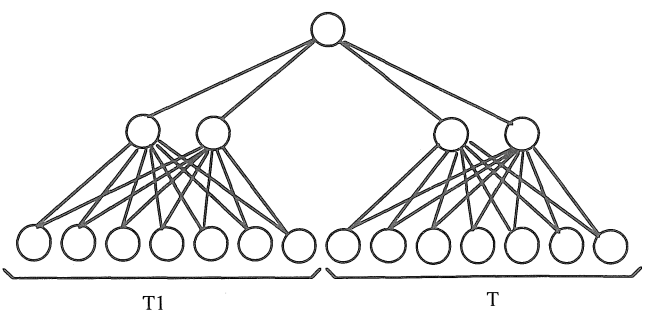


Figure 1

Table 2 presents the percentage of accuracy obtained testing the net with all the testing sets after being trained with all the training sets.

The notation N_DBi and N_DBRi identifies the net trained with the training sets from databases DBi and DBRi respectively. The notation T_DBi and T_DBRi denotes the test sets from databases DBi and DBRi respectively.

	T_DB1	T_DB2	T_DBR2	T_DB3	T_DBR3	T_DB4	T_DBR4	average
N_DB1	89.00	86.30	87.40	80.00	82.50	75.31	82.59	85.39
N_DB2	89.38	86.32	87.84	82.10	83.45	78.65	83.65	86.42
N_DBR2	88.65	85.35	87.84	80.97	83.09	76.91	82.41	85.20
N_DB3	89.39	85.07	86.86	81.84	83.98	78.75	82.41	84.22
N_DBR3	88.01	83.71	85.88	79.80	82.38	76.87	81.52	84.32
N_DB4	87.60	83.90	84.90	81.02	82.74	78.00	81.70	83.16
N_DBR4	88.35	84.75	86.47	79.63	81.49	76.61	82.59	84.53

Table 2

It is interesting to notice how for the different tests the net gave results which were relatively independent of the set it had been trained with.

This supports the belief that the net is learning to recognize the matching relation among terms: for example N_DB2 performs almost equally well when tested with T_DB1 or T_DBR4 though the tests come from databases with different statistical features.

We decided to analyze how errors were related to the number of variables and to the length of the terms involved.

The distribution of the error with respect to the number of variables and the length of the terms T_1 and T in the testing pairs was studied for every N_DBi, N_DBRi, and T_DBi, T_DBRi.

Fig. 2 shows how the error is distributed with respect to the number of variables (a) and the length (b) of the terms T_1 and T in the testing pairs separately, for N_DB2 and all the test sets.

The results of the net for the different tests were very similar. From this we can see that the degree of difficulty of the problem did not seem to be strongly related to the number of variables involved.

On the other hand the net did seem to encounter serious difficulties when the term T became large, nearly all the errors being found when such term has a length above 5.

Also an increasing degree of difficulty is related to longer T_1 terms, but not as strongly as to the length of T terms.

The others N_DBi, N_DBRi gave very similar results to the one shown for the N_DB2 case.

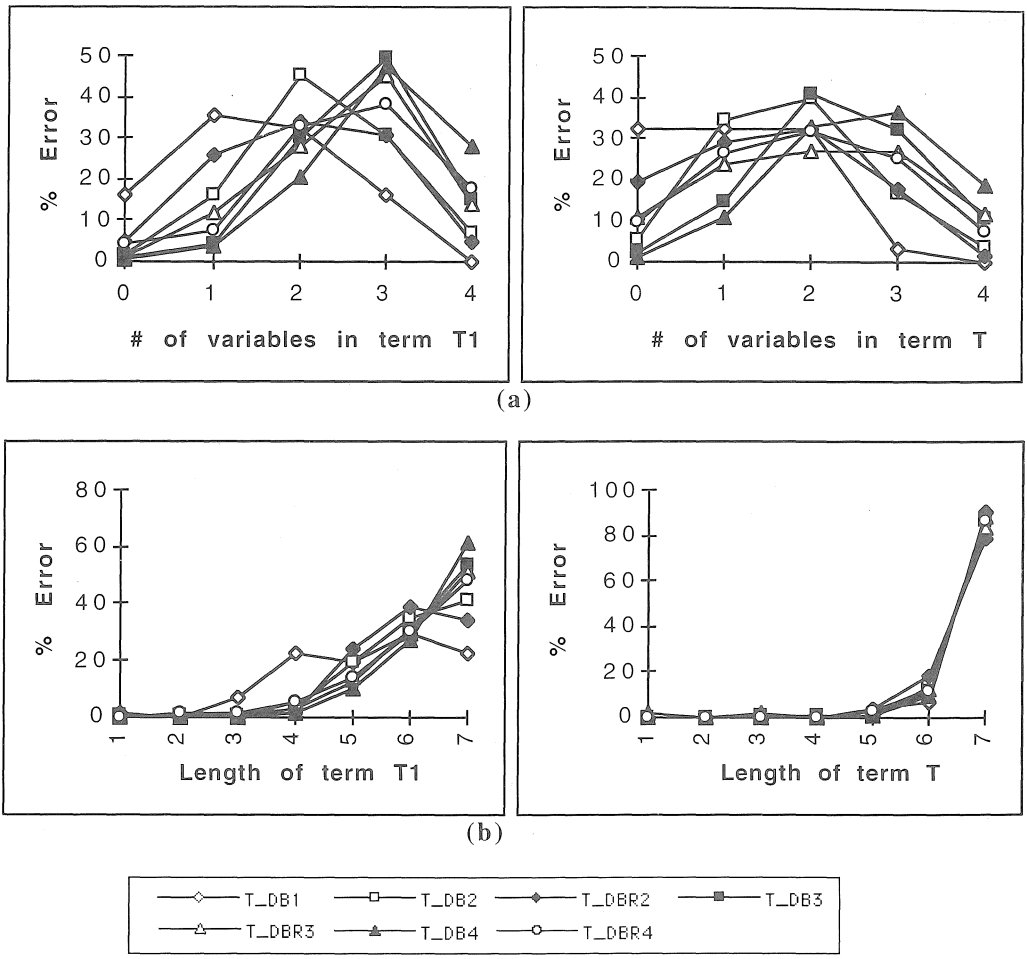


Figure 2

4. RECOGNIZING MATCHING IN THE GENERAL CASE

We then moved to the final goal stated at the beginning of this paper: to identify a class of nets that would work independently with respect to the number n of terms to be verified for matching against the term T.

Notice that increasing the number of input terms would lead to nets of increasing complexity (in terms of the number of connections); this would soon become impractical for training on the small machines (PC 486) we have available.

In order to solve the problem for higher values of n, larger nets were built utilizing the net which was used for solving the problem for n=1 as a building block.

The resulting net architecture for any value of n is shown in Fig. 3

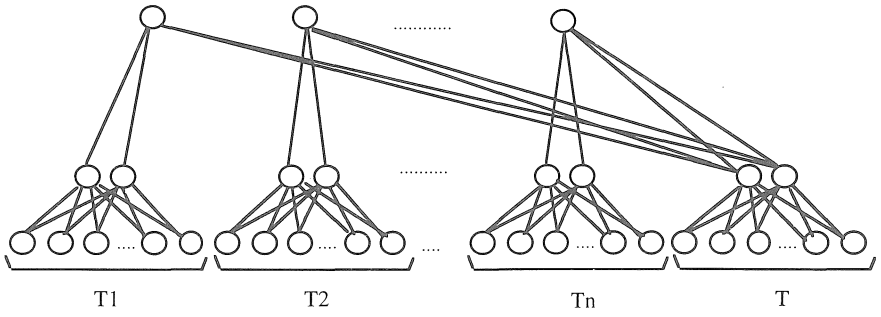


Figure 3

The connections from the T_is to the hidden layer and then to the output layer were assigned the same value as the respective connections from T_1 in the previous net after training. In the same way, connections from T to the hidden layer and then to the output layer were assigned the same value they had in the respective connection of the previous net after training.

It is important to notice how such bigger nets would not therefore have to be trained, thus avoiding the problem mentioned in the beginning of this section.

In the experiments presented in the following the values of the connections of N_DB2 were assigned to the connections of the new nets.

Such nets were then tested for problem(1) with varying n, using (n+1)-tuples of terms randomly built using terms found in DB4. The percentage of accuracy obtained for n=4, 8, and for a maximum length of the term T varying from 4 to 7 are shown in table 3.

	n = 4	n = 8
max. length 7	72.66	60.9375
max. length 6	90.716	81.1358
max. length 5	96.15	92.1154

Table 3

As expected, the percentage of accuracy decreases as n increases, but still the nets are quite reliable as long as the term T has a maximum length of 5.
It is worth to point out how a sequential solution of the problem would require an increasing amount of time as the number of terms to be verified for matching against a subterm of term T increases. On the other hand the amount of time using the approach proposed in this paper remains constant.

5. CONCLUSIONS

The purpose of this project was to build a Neural Network that would receive as inputs (n+1) terms $\{T_1,\ldots,T_n,T\}$ and would suggest which T_is are matched by a subterm of T, with varying n.
The nets presented in this paper can solve the problem for n=1 with a good degree of accuracy. These nets were then used as a building block for larger nets that were applied to solve the problem for higher values of n.
The greatest shortcoming of the method is constituted by the fact that the performance of the nets decrease for increasing size of n. The second part of this paper presents a way to improve these results, by specializing different nets to handle those cases which were found to be particularly difficult for the nets introduced here.

REFERENCES

C.L.Chang, R.C.T.Lee [1973], "Symbolic Logic and Mechanical Theorem Proving", Academic Press, New York

N.Dershowitz [1989], "Completion and its applications", in Resolution of equations in algebraic structures, pp. 31-48.

J.Hsiang, M.Rusinovitch [1987], "On Word Problems in Equational Theories", 14th Int. Conf. on Automata Languages and Programming, Karlsrhue.

J.M.Hullot [1980], "A catalogue of canonical term rewriting systems", Rept. CSL-113, SRI International, Menlo Park, CA.

D.E.Knuth, P.Bendix [1970], "Simple Word Problems in Universal Algebras", Proceedings of the Conf. on Computational Problems in Abstract Algebras, Pergamon Press, pp. 263-298.

A.Paccanaro , C.Willigs[1994], "Guiding Term Simplification in a Rewriting Based Automated Theorem Prover through a Neural Network", Proceedings of the XIV Int. Conf. of the Chilean Computer Science Society, pp. 315 - 322

A.Paccanaro [1995], "Guiding Term Simplification using Neural Networks: some results for the Group Theory", Proceedings of Neural, Parallel and Scientific Computations, v. 1, pp. 377 - 382

DISEÑO E IMPLEMENTACIÓN DE REDES HÍBRIDAS NEURO-DIFUSAS PARA EL CONTROL DE PROCESOS INDUSTRIALES *

Ing. Hernán Darío ALVAREZ, M.Sc. & Ing. Demetrio Arturo OVALLE C., Ph.D

UN-GIDIA (Grupo de Investigación y Desarrollo en Inteligencia Artificial, Reconocimiento de Patrones e Informática Educativa)
Postgrado en Ingeniería de Sistemas
Facultad de Minas

UNIVERSIDAD NACIONAL DE COLOMBIA, Sede Medellín,
Calle 65 # 78-28, Bloque M1 piso 4°
tels: (94) 425 98 29 / 45, fax: (94) 234 10 02
e-mails:hdalvare@perseus.unalmed.edu.co,
dovalle@perseus.unalmed.edu.co

RESUMEN

La utilización de técnicas híbridas de Inteligencia Artificial se presenta como opción viable para abordar la construcción de sistemas inteligentes, con muchas ventajas sobre las individuales. Se propone una Red Híbrida Neuro-Difusa (RHND) que, sin utilizar una herramienta de Sistemas Expertos, implanta un control difuso con cualquier número de reglas de producción, trasladando el significado de dichas reglas y su relaciones internas a las conexiones de una red que posee nodos con funciones particulares. Se examina entonces la aplicación de una RHND al control en la producción de carbón activado de una planta piloto ubicada en la UNAL-Medellín. Finalmente, se exponen los resultados comparativos entre el controlador tradicional y el implantado con la RHND, para este proceso, comprobándose las ventajas del controlador que usa técnicas de IA para su operación.

1. INTRODUCCIÓN

Cualquiera que sea el área de la ingeniería en la que se esté intentando obtener una modelación de un sistema físico, lo primero que se debe abordar es la selección de un mecanismo de representación de todo el conocimiento que se tiene sobre el fenómeno. Los medios de representación se pueden dividir en dos grandes grupos.

1.1 Representaciones Simbólicas

Son las más comunes, puesto que atienden al mecanismo natural de comunicación del ser humano, la representación de los eventos de la naturaleza mediante símbolos, más o menos universales, que sin ser el objeto mismo que representan, transmiten el mensaje de la existencia

* *Esta investigación es patrocinada por el proyecto 1118-14-091-94 de Colciencias intitulado: "Diseño y Desarrollo Metodológico de SBCs (Sistemas Basados en Conocimientos) distribuidos y cooperantes aplicados a la Ingeniería"*

de dicho objeto y sus características más relevantes. Esta representación presenta muchas variantes, todas con el elemento común del símbolo como substituto del objeto o propiedad real.

- **Representación Simbólica como Colección de Conocimientos**

Este tipo de representación busca emplear una estructura conceptual para describir y modelar los elementos del conocimiento que intervienen en el dominio de la aplicación y que son claves para la resolución del problema. De otra parte, cada estructura tiene asociado un proceso interpretativo o mecanismo de razonamiento el cual maneja los conocimientos almacenados en la estructura. La representaciones simbólicas más utilizadas son las reglas de inferencia, los marcos, las redes semánticas, con manejo de incertidumbre (mediante lógica difusa). A continuación se examinará esta última representación, la cual sirve de fundamento a las Redes Híbridas Neuro-Semánticas.

- **Representación Simbólica con manejo de Incertidumbre (mediante Lógica Difusa).**

En la lógica tradicional la pertenencia de un elemento a un conjunto se define con un atributo de pertenencia o no pertenencia, valores absolutos únicos y excluyentes uno del otro. Con este tipo de conjuntos claramente delimitados, la representación del conocimiento de un experto, que por lo general se expresa con términos lingüísticos, resultaba muy difícil de capturar con toda su riqueza, por lo que se recurrió a definir una clase particular de conjuntos de elementos que no tienen un límite definido sino difuso, de modo que para valorar la pertenencia de un elemento a uno de estos conjuntos borrosos se recurre a un calificativo numérico continuo entre 1 y 0 (pertenece o no pertenece respectivamente), denominado factor de pertenencia, que permite representar términos lingüísticos como Alto, Bajo, Rápido, Despacio, etc. como subconjuntos difusos que se superponen para conformar un universo de discurso del problema (Self, 90), (Driankov et al, 93).

1.2 Representaciones Conexionistas

Estas técnicas aparecen como imitaciones burdas del comportamiento del cerebro de los mamíferos, en los cuales, millones de células, muy limitadas en términos de la función individual que realizan, al unirse en una red altamente interconectada logran desarrollar tareas bastante complejas. Precisamente, el almacenamiento del conocimiento sobre el fenómeno se hace en las conexiones que unen los nodos individuales de la red, además de contar con la posibilidad de operar sobre la función de procesamiento que realiza cada nodo individualmente. En este tipo de representaciones, el símbolo no es evidente, puesto que la fortaleza de las conexiones y los parámetros internos de la función del nodo no son de interpretación directa, por lo que algunos autores denominan este tipo de representación como de caja negra. A diferencia de la representación simbólica, en la representación conexionista, los pesos de las conexiones y los parámetros internos de cada nodo se deben entrenar, con base en ejemplos o muestras del comportamiento del sistema que se desea modelar. Este tipo de procedimientos atienden a obtener muy buenos datos de entrada y salida del sistema. Existen varios tipos de medios para representación conexionista.

- ***Redes Neuronales Artificiales.***

Es tal vez la tecnología con más carga de emotividad cuando se la ataca o cuando se la defiende, tal vez porque su idea original está en el cerebro del hombre, puesto que con las redes neuronales

artificiales se busca mimificar el comportamiento de las redes neuronales naturales que residen en el cerebro de los mamíferos. Se han logrado modelos matemáticos que permiten simular el comportamiento de una neurona individual frente a la carga de información que recibe desde otras neuronas y la manera en que produce una respuesta de inhibición o activación con esta información (Freeman and Skapura, 93). Con estos modelos individuales ha sido posible construir redes que burdamente imitan las reales del cerebro, de modo que se logra procesar información con un mecanismo similar al natural, logrando alta velocidad y características típicamente humanas como la generalización y el aprendizaje a partir de ejemplos, además de obtener procesamiento de información incompleta, vaga o incluso incierta, con la certeza que lo hace un ser humano. De estas características se han aprovechado áreas como la aeronáutica para construir y entrenar RNA que puedan operar un avión a altas velocidades (3 o más veces la velocidad del sonido) como la hace un ser humano a bajas velocidades, límite natural de la velocidad de reacción del hombre. Se han ensayado RNA que imitan perfectamente las destrezas del mejor piloto, pero que corriendo en un computador a alta velocidad de procesamiento, pueden operar el avión a velocidades mucho mayores que el mejor piloto humano.

Existen diversas configuraciones de RNA que se han ensayado, algunas buscando explicar el comportamiento del cerebro humano en labores como el razonamiento, el aprendizaje, etc. otras simplemente como invenciones ingenieriles de aplicación práctica. Pueden mencionarse, entre otras: la RNA de Alimentación desde Adelante (Feedforward Networks), la RNA de Hopfield o la RNA de Kohonen. Dichas redes no son el objeto de este artículo y por lo tanto no serán examinadas. Información más detallada sobre ellas puede ser encontrada en (Alvarez & Ovalle, 95).

- ***Redes Híbridas Neuro-Semánticas.***

La configuración en red, con nodos y enlaces entre ellos, permite más riqueza estructural y paramétrica que las representaciones simbólicas. A pesar de esta ventaja, las redes de nodos con funcionalidades genéricas: sigmoides, discriminantes duros, etc. pueden aprovechar en muy poca magnitud el conocimiento previo del fenómeno o la experiencia que sobre el mismo se tenga, porque estos nodos no permiten modificar el tipo de función de activación o de procesamiento interno. Por lo anterior, se requiere de la introducción de funciones de tipo semánticas, como calificativos lingüísticos de eventos, de estados posibles, de tendencias de variación, etc. y el aporte adicional del establecimiento de conexiones ponderables entre dichos nodos (Lin and Lee, 91). El aporte que realizan las redes neuronales artificiales a este tipo de representaciones, es significativo en la estructura y configuración genérica de los nodos, así como en los métodos de ajuste de los ponderadores o pesos de las conexiones, conocidos como entrenamiento o aprendizaje. La representación semántica, por su parte, aporta a esta representación híbrida la posibilidad de usar a-priori conocimiento y experiencia sobre el fenómeno que se está representando. Este tipo de aporte semántico se constituye en una pista o indicio para facilitar el aprendizaje de la red. Adicionalmente, se tienen ventajas de tipo funcional y operativo, porque se logra disminuir el tiempo de entrenamiento y se facilita el diseño inicial de la red, además permite representaciones mejores de sistemas complejos, a un costo computacional menor y con un grado de generalización más alto.

Una de las aproximaciones que más se está utilizando en el mundo por su versatilidad, en este tipo de configuraciones, es la de atributos difusos expresados como reglas típicas:

SI <atributo difuso> **ENTONCES** <atributo difuso>

El grupo de reglas que se obtienen como representación del conocimiento particular sobre el fenómeno, pueden usarse para inferir mediante algún mecanismo deductivo, consecuencias de las evidencias disponibles (Sugeno and Park, 93). Aunque lo más normal es la implantación mediante un motor de inferencia sobre las reglas, escritas como líneas individuales, es posible representar ese grupo de reglas como una red de nodos y realizar la inferencia como una propagación de información desde la entrada de la red hasta su salida. Las redes constituidas de esta manera, denominadas Redes Híbridas Neuro-Difusas RHND, toman su topología de las RNA típicas, pero intercalan capas de nodos con funciones internas más complejas, como por ejemplo las funciones de pertenencia a subconjuntos difusos. Esto no implica que se pierdan las habilidades típicas de las RNA en los aspectos de generalización y aprendizaje, ganando además en velocidad de entrenamiento, puesto que en este caso el número de pesos (parámetros) a ajustar es mucho menor, con la ventaja adicional de que se parte con un conocimiento heurístico del proceso, lo que facilita sobremanera el diseño del sistema en las etapas iniciales. Además, se pueden usar todas las reglas posibles de la partición de los universos de discurso de entrada, sin la reducción en el número de reglas recomendada en la literatura (Li and Lau, 89), (Yamakawa, 93). Esto permite que las transiciones en la salida del modelo sean más suaves y continuas, sin aumentar la carga computacional, debido al esquema de ejecución en paralelo que se logra con la RHND.

El ajuste de los parámetros de una RHND se puede hacer supervisadamente, mediante cualquier técnica de entrenamiento de redes: Algoritmos Genéticos (Holland, 93), Transmisión Química Bacteriana (TQB o en ingles Bacterial Chemotaxis) (Bremermann and Anderson, 90) e incluso retropropagación del error con algunas modificaciones (Fukuda and Shibata, 92). Cualquier técnica que se use debe poder acceder, sin distingo, a todos los parámetros ajustables de la red, que aquí a diferencia de las RNA son mixtos: internos a los nodos y externos a estos, como ponderaciones de conexiones.

2. CONTROL DE PROCESOS INDUSTRIALES

El control del estado de las variables de cualquier proceso productivo es una labor irremplazable, cuando se desea obtener la mejor calidad del producto final y aumentar la utilidad económica. Adicionalmente, los requerimientos de preservación ambiental están exigiendo más controles y más sofisticación en el manejo del proceso. Todo esto conduce a la necesidad de sistemas de control cada vez más efectivos, en las tareas bastante complejas de algunos procesos industriales, hasta hoy controlados pobremente. Muy a menudo una variable controlada en un proceso es una variable de carga para el siguiente en la línea de producción, lo que obliga a su control. Debido a que las variables de carga y las variables manipuladas afectan a las variables controladas, el problema del control es mantener una relación adecuada entre las variables de carga y la variables manipuladas, de modo que el proceso presente un comportamiento prefijado en sus variables controladas. El trabajo del sistema de control es, por lo tanto, permanentemente determinar y actualizar los valores de las variables manipuladas, con dos posibles fines: Mantener las variables controladas en los

valores deseados o puntos de ajuste, denominado control regulatorio; Alcanzar un valor deseado de la variable controlada, denominado control servo o control de seguimiento (Tracking Control).

Existen diversas maneras de implantar un sistema de control que actúe sobre el proceso que se desea manejar. Las más tradicionales y antiguas son todas mecánicas, evolucionando luego a las hidráulicas, posteriormente a las neumáticas, las cuales todavía sobreviven luego de reinar por más de medio siglo, hasta llegar a las tecnologías de controladores eléctricos y electrónicos, los cuales desde la irrupción del circuito integrado microprocesador en la década de los setenta, originaron el control asistido por computador, que es la implantación de mecanismos de control que más ha avanzado en los últimos cuarenta años, fundamentalmente debido a la flexibilidad que brinda en el cambio del control mismo en términos de modo, estrategia y aplicación.

No obstante el avance logrado con la irrupción del computador como herramienta para realizar control sobre el proceso, existen una buena cantidad de procesos que no han podido automatizarse usando técnicas tradicionales y algunas algorítmicas avanzadas. Esto se debe principalmente a las características inherentes al proceso como no linealidades, interacción fuerte entre variables, variaciones con el tiempo de las características del fenómeno, etc. El grado en el cual se requiere la intervención humana en el control de un proceso es un índice de su complejidad o de la baja capacidad de control del sistema instalado para este fin (Kinney, 83; Badavas, 84).

Una manera de hacer control que se esta abriendo paso rápidamente en el mundo para estos procesos, hasta hoy solo bajo control manual, es el Control Inteligente. Con este tipo de control se busca imitar las habilidades de un operador humano, que por lo general es el ultimo recurso y el que siempre logra un buen desempeño en la tarea de control sobre estos sistemas complejos. Estas implantaciones gozan de características de inteligencia, por lo que es mediante el uso de técnicas de Inteligencia Artificial que se puede llegar a este tipo de controladores inteligentes.

3. UNA PROPUESTA DE RED HÍBRIDA NEURO-DIFUSA APLICABLE AL CONTROL DE PROCESOS INDUSTRIALES

El uso de herramientas individuales de IA aplicas al control de procesos industriales es hoy un tópico bastante estudiado. Los que más avance han logrado son las RNA y los SBCs. Estos últimos son sistemas que se caracterizan por tener una representación de conocimientos sobre un área particular del saber. En control, han encontrado gran uso debido a su facilidad para emular el comportamiento humano frente a un problema particular (conocimiento experto) y permitir la mejora continua en el conocimiento almacenado. Una primera aproximación al control asistido por computador, que se apoya en esta técnica, es el uso de reglas exactas, sobre las cuales no pesa ninguna clase de incertidumbre. Por lo general esto no funciona del todo bien, puesto que sobre la información suministrada por el grupo de expertos operarios pesan varios tipos de incertidumbre. Aparecen enfoques alternativos, entre los que vale mencionarse la Lógica Difusa (Fuzzy Logic), un enfoque totalmente nuevo y que supera el escollo lingüístico en la comunicación con el grupo de expertos, además de proporcionar un mejor entendimiento de las cláusulas del sistema.
En esta Lógica Difusa (Zadeh, 65), cada término lingüístico calificativo de una evidencia, forma un subconjunto en el que la pertenencia de los elementos está en todo un intervalo de infinitas posibilidades (0 a 1), definido mediante un subconjunto difuso de grados de pertenencia μ, a un

calificativo lingüístico particular, sobre todo el universo de discurso o intervalo total de la variable. Con esta inclusión de una lógica difusa, para el manejo de las reglas, se gana también capacidad de aprendizaje, la que en los sistemas simples se logra mediante la generación de nuevas reglas o abolición de reglas existentes, con el riesgo de perder la consistencia del sistema experto. En el enfoque difuso, el aprendizaje se puede obtener con la modificación de la forma misma de los subconjuntos difusos de pertenencia a cada calificativo lingüístico, lo que permite aplicar técnicas de retropropagación de alguna señal, indicativa de la calidad del control para adecuar la forma de dichos subconjuntos (Fukuda and Shibata 92), (Werbos, 93).

De otra parte, las RNA han encontrado un terreno propicio para producir neurocontroladores capaces de mimificar el comportamiento de un operador humano en tareas tan complejas como el pilotar un avión de combate, todo mediante sus grandes capacidades de aprendizaje y generalización a partir de ejemplos. Esto pone un punto bastante alto pues solo es necesario contar con información suficiente sobre el fenómeno (Valores de entrada y salida por lo general) parra lograr que una RNA con la topología adecuada pueda imitar con lujo de detalles el comportamiento del mejor operador humano, e incluso realizar labores para las cuales nunca fue entrenada, o sea de las que no habían datos de ejemplo.

En el mundo del control asistido por computador, la introducción del neurocontrol y del control difuso (fuzzy control), ha generado una gran cantidad de desarrollos en ambas tecnologías, todos con la mira del control autónomo, paso último en el ideal del mando automático de un proceso industrial, pero que exige el dominio previo de las técnicas de control inteligente. Se empieza a ver como horizonte en este campo el uso de técnicas híbridas, que resalten las ventajas y soslayen las desventajas de las técnicas individuales. Una aproximación que muestra gran futuro son las redes híbridas, conformadas por nodos en los cuales se realizan funciones diferentes sobre los datos de entrada, de modo que al combinar sus salidas en red, se logra una función con alto grado de no linealidad, con características particulares de acuerdo con las funciones internas de dichos nodos.

Con esta visión, la manera tradicional de implantar un control difuso, mediante una herramienta o entorno (shell) de desarrollo de sistemas expertos, está cambiando, debido a la exigencia de más facilidad de sintonía, adaptación y aprendizaje en estos sistemas de reglas de producción.

Entre los inconvenientes más reportados en los sistemas de neurocontrol se pueden destacar:

- Gran cantidad de tiempo en la selección y preparación de los datos de entrenamiento, debido al gran número que se requiere para lograr buena generalización en la red entrenada.
- Mucho tiempo en el proceso de entrenamiento, aun con computadores de alta velocidad. En ocasiones decenas de horas para una red relativamente pequeña.
- La necesidad de entrenar la red con todos los patrones nuevamente, cuando se quiere cambiar un patrón o un punto de operación dados, con los consabidos costos en tiempo de entrenamiento.

De otro lado, el control difuso adolece también de inconvenientes cuando se enfrenta a la tarea de control, con miras a ser un mecanismo inteligente:

- Perdida de inteligibilidad en la base de conocimiento (por ejemplo un conjunto de reglas), cuando se deja libre para que un módulo de aprendizaje la modifique.
- Poca flexibilidad en los programas de implantación de sistemas expertos, cuando se desea cambiar un valor particular en una regla.
- Perdida de consistencia en la base de conocimiento cuando las reglas se modifican individualmente a la luz de una labor de aprendizaje.

No se mencionan aquí las ventajas que presentan ambas técnicas sobre otras tradicionales que han intentado abordar la labor de aprendizaje (por ejemplo con técnicas algorítmicas), puesto que diversos artículos de la literatura muestran implantaciones no híbridas que operan con éxito (Li, Y. and Lau, C.C., 89),(Yamakawa, Y., 93), (Karr, C. and Gentry, E.J., 93).

Con el fin de evitar los inconvenientes de las técnicas individuales y reforzar sus ventajas, se han propuesto varios modelos de sistemas híbridos (Lin, C and Lee, C.S. 91), (Werbos, P., 93), de los cuales es posible tomar aportes para construir un modelo que mezcle las habilidades de entrenamiento de las RNA, con el manejo de información bajo incertidumbre del razonamiento difuso, de modo que se construya una manera mucho más eficiente de realizar aprendizaje en línea y desarrollar la tarea de control inteligente.

3.1 ¿Qué Son Las Redes Híbridas Neuro-Difusas (RHND) ?

Estas redes toman su topología de las RNA típicas (Freeman, J.A. y Skapura, D.M., 93), con nodos que desarrollan una función simple (Sigmoide, Logística o Lineal), pero intercalan capas de nodos con funciones internas más complejas (funciones de pertenencia, implicación y concreción en subconjuntos difusos). Esto no quiere decir que se pierdan las habilidades típicas de las RNA en los aspectos de generalización y aprendizaje, ganando además en velocidad de entrenamiento, puesto que en este caso el número de pesos (parámetros) a ajustar es mucho menor, con la ventaja de que se parte con un conocimiento heurístico del proceso, lo que facilita sobremanera el diseño del sistema en las etapas iniciales.

Como ejemplo, la Tabla 1 muestra la matriz de reglas de un control difuso simple. La Tabla 1 muestra la RHND que implanta este control difuso, usando todas las posibles combinaciones de las variables de entrada del proceso tomadas como universo de discurso (Error e y cambio del Error con el tiempo Δe) y usando una salida como cambio en la potencia aplicada ΔP.

Tabla 1. Matriz de reglas para un control difuso simple, tipo PI.

	e es Negativo N	**e es Cero Z**	**e es Positivo P**
Δe es Positivo P	Regla 1, ΔP es Z	Regla 2, ΔP es P	Regla 3, ΔP es N
Δe es Cero Z	Regla 4, ΔP es P	Regla 5, ΔP es Z	Regla 6, ΔP es N
Δe es Negativo N	Regla 7, ΔP es P	Regla 8, ΔP es N	Regla 9, ΔP es Z

3.2 Implementación de la Red Híbrida Neuro-Difusa.

La red que se propone, consta básicamente de 5 capas, cada una con un tipo particular de nodo genérico, de modo que se realiza toda la operación de un control difuso como si fuera una red. La

función y característica de los nodos en cada capa se muestran para un control difuso tipo PI, aunque puede usarse para otros tipos de control difuso (Driankov et al., 93).

La Capa 1 o capa de entrada es la encargada de cualquier preprocesamiento de las entradas que se estime necesario. La función que desarrollan estos nodos, en el modelo propuesto, es un acotado y escalado de los valores provenientes de sensores, de modo que los parámetros característicos del nodo son el valor central y el ancho del intervalo de la variable. Todos los universos de discursos quedan de este modo normalizados a valores de [-1, 1] ó [0,1].En esta capa, los parámetros a entrenar no son los pesos de las entradas (pues estos siempre valen uno), sino los valores del centro y del ancho del intervalo de la variable que se trabaja.

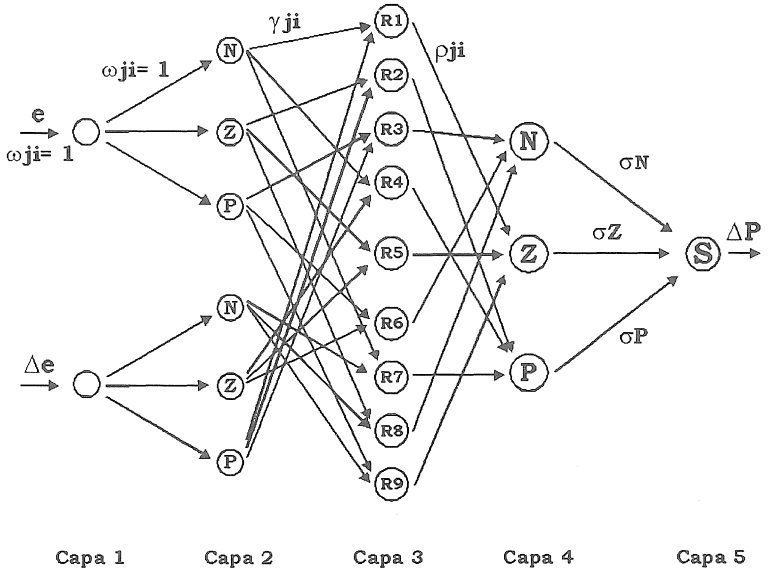


Figura 1. Red Híbrida Neuro-Difusa que implanta el control difuso de 9 reglas del ejemplo.

La capa 2, denominada capa de difusión o de cálculo de factores de pertenencia μ, es la encargada de realizar la difusión (Fuzzification) de los valores numéricos acotados y normalizados, para entregar en la salida de cada nodo un valor μ en el intervalo [0,1], que es el valor de pertenencia de la entrada al conjunto difuso que representa el nodo. Entre estos nodos existen diversos tipos particulares, de acuerdo con la forma que se le de al conjunto difuso, por ejemplo: Triángulos, Campanas Gaussianas, Trapecios, etc. Cada una de estas figuras presenta un número diferente de parámetros, los cuales son los valores a ajustar en esta capa, porque aquí de nuevo los pesos de las conexiones ωji son iguales a uno.

La capa 3, es la capa que contiene la primera parte de la base de conocimientos. En esta capa cada nodo calcula el antecedente neto de una regla de producción del tipo:

SI <proposición difusa antecedente p> **ENTONCES** <proposición difusa consecuente q>

donde generalmente la proposición difusa antecedente es una función lógica **Y** entre dos o más grados de pertenencia a conjuntos difusos diferentes. De este modo, la función interna que desarrollan estos nodos de reglas, depende del tipo de T-Norma que se use para concluir sobre la

proposición Y. Nótese que cualquier proposición difusa que combine O, Y, NO (OR, AND y NOT), puede convertirse con facilidad en dos o más reglas simples con conectivo único tipo Y. La salida de cada nodo en esta capa es un factor de pertenencia μ que refleja el resultado de la operación lógica entre las entradas. Por ejemplo si se usa una T-Norma de mínimo, la salida del nodo será el mínimo de todos los valores de entrada.

En los nodos de la capa 3, los parámetros de entrenamiento son los pesos de las conexiones entre estos nodos y los nodos de difusión, puesto que los parámetros internos no deberían modificarse, porque se cambiaría drásticamente la T-Norma que se aplica, aunque queda abierta esta posibilidad de cambio. P. Werbos ha llamado a los pesos de las conexiones entre capa 2 y capa 3 Elasticidades γji , y muestra como variando estos valores, es posible dejar intactas las reglas como relación entre antecedente y consecuente, lo que permite preservar la inteligibilidad del sistema, incluso después de ser sometido a un proceso de entrenamiento (Werbos, P., 93). Una interpretación a la luz del efecto final de estos pesos, que no es más que reforzar un miembro del antecedente frente a los otros, podría ser que estas elasticidades se relacionan con un cierto tipo de meta-conocimiento, que el operador humano ejecuta, pero que no puede detectarse con facilidad mediante técnicas de ingeniería del conocimiento y que está en los datos de entrada y salida del sistema. Debe recalcarse como estos nodos de reglas de la capa 3, no realizan la inferencia de cada regla. Ellos, solo realizan el calculo del factor de pertenencia de la proposición difusa antecedente. Pero, la conexión entre cada nodo de regla y la capa siguiente, representa la regla completa.

La capa 4, denominada capa de inferencia o de conjuntos del consecuente, es la encargada de tomar los valores de salida de los nodos de regla de la capa anterior, y mediante el uso de uno de lo criterios comunes de interpretación de la implicación $p \rightarrow q$ evalúa la salida de la regla total:

SI p **ENTONCES** q

Las entradas a cada nodo en esta capa, están ponderadas mediante pesos que modifican la participación que tiene el resultado del antecedente difuso, sobre la conclusión final que tendrá dicha regla. Estos pesos ρji , del mismo modo que las elasticidades γji , actúan como meta-conocimiento, que puede ser modificado durante el aprendizaje. En los nodos de esta capa, los parámetros de entrenamiento son los pesos ρji de las conexiones que llegan de la capa anterior. Los parámetros internos de cada nodo no se deben modificar, pues hacerlo causa un cambio de la implicación que se usa, además de acuerdo con el tipo de concreción que se haga en la capa 5, uno de los parámetros de cada nodo puede ser el centro de los conjuntos difuso de llegada o valor lingüístico de la salida recomendada al Elemento Final de Control (EFC).

Por último, la capa 5, salida en esta RHND, es la encargada de realizar la concreción (Deffuzification), que permite convertir el conjunto difuso que resulta de la inferencia de todas las reglas, en un valor numérico, que en última instancia es el que se aplica al EFC del proceso. En los nodos de esta capa, se puede aplicar cualquier método de concreción, de las muchos que se han planteado al momento y dependiendo del que se escoja, la ponderación de las conexiones que llegan a estos nodos σji , pueden ser los centros de los conjuntos difusos del consecuente o estar fijos a uno, siempre que estos centros queden como parámetro interno en los nodos de la capa 4.

Vale la pena resaltar que el entrenamiento puede modificar los valores de estos centros en cualquiera de los dos casos.

3.3 Entrenamiento de la Red Híbrida Neuro-Difusa

Sin importar la topología interna que se use, siempre que se respete el desempeño del control difuso en la red híbrida, es posible disponer de un buen número de parámetros de aprendizaje, sobre los cuales recae la tarea de "Representación" y almacenamiento del conocimiento, ósea el aprendizaje. El ajuste de estos parámetros se puede hacer supervisadamente, mediante cualquier técnica de entrenamiento de redes: Algoritmos Genéticos (Holland, J.H. 75), Transmisión Química Bacteriana (Bacterial Chemotaxis, Bremermann, H.J. and Anderson, R.W., 90) e incluso retropropagación del error con algunas modificaciones. Cualquier técnica que se use debe poder acceder sin distingo a todos los parámetros ajustables de la red, que aquí a diferencia de las RNA son mixtos: internos a los nodos y externos a estos como ponderaciones de conexiones.

Es importante mencionar el significado que puede darse a los parámetros ajustables, al clasificarlos como parámetros de ajuste grueso (aprendizaje) y parámetros de ajuste fino (Adaptación o sintonía). Esta diferenciación es importante cuando se aspira a construir un control con características inteligentes, porque no todos los cambios del proceso son permanentes o de una duración temporal suficiente para provocar un cambio radical en las características y constitución del controlador (aprendizaje), y por el contrario, pueden ser corregidos con una sintonía o ajuste fino sobre los parámetros que permitan cambios leves (sintonía y adaptación).

3.4 Aplicación de la Red Híbrida Neuro-Difusa

La red propuesta fue desarrollada para implantar el control difuso de un reactor de lecho fluidizado, denominado Activador, en el cual se realiza la gasificación remanente y activación del carbón molido, después del proceso de gasificación principal en un reactor precedente, denominado Pirolizador. El activador hace parte de la planta piloto para la obtención en continuo de carbón activado en lecho fluidizado, método en proceso de patente por parte del grupo de investigación en Carboquímica de la Facultad de Minas de la Universidad Nacional de Colombia, sede Medellín (Aguirre y col., 91). En este tipo de procesos, la diversidad de reacciones químicas que ocurren, además de los cambios súbitos en las variables de carga (carbón alimentado, flujo de gases de fluidización, etc.), hacen que un control tradicional PID no pueda manejar con eficiencia la globalidad del proceso: Pirolización y Activación en serie, e incluso se quede corto en el control individual de cada reactor.

El proceso se simuló mediante balances de materia y energía, así como correlaciones experimentales de diverso autores sobre el fenómeno reactivo y de transferencia de masa. Para la simulación y puesta en operación de la RHND en un computador secuencial, se desarrolló un código en C++, el cual permite la definición de nodos con funciones particulares, pero que dispone de 4 tipos base, de los cuales se desprenden subtipos relacionados. Estos tipos base son:

- Tipo 0, Nodos funcionales (capa 1 y 4), que aplican una o más funciones.
- Tipo 1, Nodos Difusos (capa 2), que entregan el factor de pertenencia μ.
- Tipo 3, Nodos de T-Normas (capa 3) para operaciones lógicas Y.
- Tipo 4, Nodos de Concreción (capa 5), que entregan un valor numérico.

Tabla 2. Resultados comparativos de los dos controladores.

Tipo de control y efecto medido.	CONTROL PI		CONTROL DIFUSO	
Perturbación en la carga del char.	*Sobre impulso en °K.*	*Tiempo Establec. en minutos.*	*Sobre Impulso en °K.*	*Tiempo Establec. en minutos.*
Aumento de 50% en flujo de char	14.8	> 30	2.2	7.0
Disminución 50% en flujo de char	15.0	> 30	1.3	5.3

Para que los parámetros de sintonía, adaptación y aprendizaje, estén disponibles con toda facilidad, el sistema almacena todos los parámetros internos y externos (pesos) de los nodos en un archivo matricial. Cuando ya se tiene operando la red, el objeto NodoRed posee funciones miembro que permiten el cambio de parámetros internos, mientras que el objeto Red posee dos funciones independientes, una dedicada a Sintonía y Adaptación y otra dedicada al Aprendizaje. Ambas pueden variar parámetros internos y externos a los nodos.

La Tabla 2 muestra el desempeño del mejor controlador PI y el control difuso implantado con una RHND, cuando se aplican diversas perturbaciones en la carga de carbón semi-procesado (Char) al activador. Este desempeño se mide en términos del sobre impulso en la temperatura del reactor y en el tiempo de establecimiento que requiere el controlador para volver al valor prefijado (en este caso 1173 °K) la temperatura del reactor.

Como puede apreciarse en la Tabla 2, el desempeño del control difuso resultó muy superior comparado con el comportamiento del mejor control PI que se puede diseñar para este reactor, lo que demuestra la superioridad de esta técnica cuando se enfrenta al control de sistemas complejos.

4. CONCLUSIONES Y RECOMENDACIONES

Entre los aspectos que hacen atractivo el uso de RHND para la implantación de controles difusos, frente al camino tradicional de usar herramientas de sistemas expertos, pueden mencionarse los siguientes:

- Capacidad de aprendizaje sin que se pierda significado de las reglas originales o se presenten distorsiones severas en la interpretación del sistema después del entrenamiento.
- Disponibilidad de algoritmos de entrenamiento supervisado, ampliamente probados en RNA, los que pueden usarse con muy pocas modificaciones en las RHND.
- Menor tiempo de inferencia frente a las implantaciones con programas de SE, si además se procesa en forma masivamente paralela, puesto que se evitan las arduas búsquedas en árboles de decisión.
- La facilidad de programación de una RHND frente a la complejidad inherente de los programas de SE. Se permite la generación automática de código con mucha más facilidad.

La principal desventaja que presenta esta aproximación es su capacidad limitada de explicación cuando se compara con la que tienen los programas de SE. Esta tarea que informa al usuario sobre el razonamiento que se utilizó para obtener una conclusión, es de importancia capital en las labores de tutoría que buscan mantener un grado mínimo aceptable de pericia en los operarios. Esta característica podría mejorarse mediante técnicas de grafos que indiquen, en un esquema de la RHND, el camino que tomó la inferencia.

Finalmente, puede decirse que este tipo de sistemas híbridos de procesamiento de información, hacen que el conjunto sea mejor que las partes, lo que trae como consecuencia la formación de estructuras con alta sinergía, que encajan muy bien dentro del marco de desarrollo de los sistemas autónomos de control.

5. REFERENCIAS BIBLIOGRÁFICAS

Alvarez, H.D. & Ovalle, D.A. Complejidad, Incertidumbre y Vaguedad. Propuestas de manejo estratégico con herramientas simbólico-conexion. de IA. Seminario sobre Caos, Complejidad y Estrategia. U.Nal Med., 1995.

Alvarez, H.D. Redes híbridas y su uso en control inteligente de procesos. Memorias del Primer Congreso Nacional de Neurocomputación. Bogotá, junio de 1995.

Aguirre, J., Ocampo, A. Y Espinel, J. Desarrollo de un proceso para producción de carbón activado. Informe de investigación, Universidad Nacional de Colombia, Sede Medellín. 1991.

Bremermann H.J. and Anderson R.W. An alternative to Back-propagation. Center of pure and applied mathematics. University of California. Internal Publication number PAM-483. Jan. 1990.

Driankov, D, Hellendomm, H and Reinfrank, M. An introduction to fuzzy control. First edition. Springer-Verlag Berlin-New York. 1993.

Freeman, J.A. and Skapura, D.M Redes Neuronales, algoritmos, aplicaciones y técnicas de programación. Primera edición en español, Addison-Wesley / Diaz de Santos. 1993.

Fukuda, T. and Shibata, T. Theory and applications of neural networks for industrial control systems. IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS. Vol. 39 No 6. Dec. 1992.

Holland, J.H. Adaptation in natural and artificial systems. Second print of original text (Michigan University 1975), for MIT press edition , 1993.

Karr, C.L. and Gentry, E.J. Fuzzy control of PH using genetic algorithms. IEEE TRANSACTIONS ON FUZZY SYSTEMS. Vol 1 No 1. Feb. 1993.

Li, Y.F. and Lau, C.C.. Development of fuzzy algorithms for servo systems. IEEE CONTROL SYSTEMS MAGAZINE. April 1989.

Lin, C. and Lee, C.S. Neural-network-based fuzzy logic control and decision system. IEEE TRANSACTIONS ON COMPUTERS. Vol 40, No 12. Dec. 1991.

Pérez, G., Rojas, G. Y Urquijo, J. Expert system for real time control of rotary cement kiln. PROCEEDINGS OF THE WORD CONGRESS ON EXPERT SYSTEMS. 1993.

Self, K. Designing with fuzzy logic. IEEE SPECTRUM. November 1990.

Sugeno, M. and Park, G. An approach to linguistic instruction based learning. INTENATIONAL JOURNAL OF UNCERTAINTY, FUZZINESS AND KBS. Vol 1, No 1. Sept. 1993.

Werbos, P.J. Neurocontrol and elastic fuzzy logic capabilities, concepts and aplications. IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, vol 40, No 2. April 1993.

Yamakawa, T. A fuzzy inference engine in non linear analog mode and its application to a fuzzy logic control. IEEE TRANSACTIONS ON NEURAL NETWORKS. Vol 4 No 3. May 1993.

RED NEURAL PARA RECONOCER MINERALES EN UNA IMAGEN DIGITALIZADA

Mauricio Solar* (msolar@toqui.usach.cl)

Francisco Watkins** (watkins@lauca.usach.cl)

Patricio Perez*** (pperez@lauca.usach.cl)

Universidad de Santiago de Chile

*Departamento de Ingeniería Informática, Facultad de Ingeniería

**Departamento de Ingeniería Eléctrica, Facultad de Ingeniería

***Departamento de Física, Facultad de Ciencias

Avda. Ecuador 3659

Fono: 56 2 778-1208

Fax: 56 2 776-3511

Resumen

Este trabajo presenta el diseño e implementación de una Red Neuronal para reconocer imágenes digitalizadas a partir de fotografías de minerales. A la entrada de la red se presenta un histograma comprimido de una sección de la imágen seleccionada por el usuario que se desea reconocer, la cual es procesada por la red diseñada, identificando uno de los seis minerales enseñados.

1. Introducción.

El procedimiento actual para el reconocimiento de minerales se realiza obteniendo información de los distintos minerales que componen una roca. Para realizar esto se preparan muestras pulidas de un cuarto de pulgada de grosor. La pulimentación de la muestra permite la obtención de superficies lisas para su análisis al microscopio. La muestra pulida se denomina *briqueta*. La briqueta se coloca sobre una cuadrícula, como una manera de discretizar la imagen al microscopio. De esta manera es posible cuantificar la cantidad de puntos que abarca u ocupa cada *grano* de mineral existentes en la muestra. Un grano es una sección de la muestra que contiene un determinado mineral, y es identificado por el experto recordando las características visuales de cada mineral.

El proceso es extremadamente lento y proclive al error, dada la excesiva dependencia de la atención del experto a cargo de la detección y contabilización de puntos.

El problema planteado se divide en dos áreas relevantes: a) Reconocimiento de mineral; y b) Contabilización de puntos en el área asociada a cada grano, con el objetivo de determinar su presencia porcentual en la muestra.

Para resolver el problema de reconocimiento de minerales se procedió a digitalizar fotografías tomadas directamente desde el laboratorio donde se realiza el análisis de las muestras. A través de un *scanner*, se generó un conjunto de imágenes digitalizadas en formato TIFF, a las cuales se les realizó un pre-procesamiento para generar un archivo con patrones de entrada para el entrenamiento de una Red Neuronal (RN) diseñada para este propósito.

Para el diseño de la arquitectura de RN [1] que identifique los patrones asociados a cada mineral, se utilizó el software *Professional NeuralWorks* bajo ambiente *SunOS*. Para el proceso de

aprendizaje se utilizó el algoritmo de *backpropagation* [3] para entrenar la RN con los patrones de entrada digitalizados previamente. Luego en la fase de operación, para el Reconocimiento de Minerales, se construyó una interfaz amistosa bajo ambiente *Windows* que utiliza la RN previamente entrenada con los patrones disponibles, permitiendo el reconocimiento automático.

El producto final es un sistema capaz de reconocer los minerales a través de la inspección sobre un monitor de computador de la imagen digitalizada de una briqueta. La entrada consiste en la selección de un área de la briqueta por parte del usuario y la salida es la identificación del mineral.

Para resolver el segundo problema de obtener la presencia porcentual de cada grano de mineral en la muestra, se implementó una opción, que permite inspeccionar exhaustivamente toda la imagen para determinar la cantidad total del mineral dentro de la imagen (en tanto por ciento).

2. Diseño de la Red Neuronal.

Para entrenar la RN se dispone de 36 fotografías a color, cada una de las cuales cubría una fracción de diferentes partes de una briqueta. En todas las fotografías se observaron regiones delimitadas que correspondían a granos de diferentes minerales (muchos de ellos componentes del cobre). Basados en la información dada por el experto, se obtuvo patrones de los siguientes seis minerales utilizados:

1	Calcopirita
2	Calcosina
3	Covelina
4	Bornita
5	Pirita
6	Energita

Se procedió a digitalizar las 36 fotografías usando un scanner a color que permite almacenarlas en formato TIFF. Se desplazó una ventana de un tamaño de 10x10 pixels sobre las imágenes, seleccionando aquellas regiones que corresponden a cualquiera de los 6 minerales antes mencionados. En condiciones constantes de captura de datos, es posible afirmar que cada uno de los minerales estudiados en una muestra digitalizada poseen una textura similar para distintas muestras. La arquitectura de RN debe ser capaz de aprender estas texturas, mediante la clasificación de sus histogramas característicos. Para esto, se descompuso esta información en 3 filtros (rojo, verde y azul) y se descubrió que comprimiendo el histograma en 23 intensidades es suficiente para discriminar los minerales.

Como entrada a la RN se tiene entonces, un histograma, que es una función de frecuencia [2]. Es decir, el histograma de una imagen digitalizada es la contabilización del número de pixel clasificados por el nivel de color existente en ella.

En la construcción de la RN feedforward, se definió como entrada una capa de 23 neuronas y una capa de salida de 8 neuronas. La neurona activada en la capa de salida identifican los minerales. Se encontró necesario incorporar 2 neuronas adicionales en la capa de salida, dado que se descubrió que existen 2 patrones diferentes para identificar la covelina, y la otra neurona para activarla cuando se está en presencia del fondo de la briqueta (no hay presencia de mineral). El hecho de identificxar el fondo es importante para determinar la presencia relativa de cada mineral en la muestra, cuando esta se recorre completamente.

Los datos del histograma se mapearon al intervalo (0,1), el cual permite trabajar en la región de mayor sensibilidad de la función de transferencia $f(z)$, llamada también función logistica, en la cual el estado de la neurona de salida j en la capa s está dado por:

$$x_j^{[s]} = f(\sum_i (w_{ji}^{[s]} x_i^{[s-1]})) \qquad (1)$$

donde

$$f(z) = (1+e^{-z})^{-1} \qquad (2)$$

$w_{ji}^{[s]}$ representa el peso de la conexión desde la neurona *i* de la capa *s-1* con la neurona *j* en la capa *s*. En la fase de entrenamiento, estos pesos se inicializaron con valores aleatorios, y posteriormente se ajustaron en forma iterativa de acuerdo a:

$$\Delta w_{ji}^{[s]} = \varepsilon^{[s]} \varepsilon_j^{[s]} x_i^{[s-1]} + \alpha^{[s]} \Delta w_{ji}^{[s]}(t-1) \qquad (3)$$

donde $\varepsilon_j^{[s]}$ es una medida del error local en la neurona *j* de la capa *s*, $\varepsilon^{[s]}$ es el coeficiente de aprendizaje de la capa *s* y $\alpha^{[s]}$ es conocido como el coeficiente de momentum.

Al realizar las investigaciones sobre el efecto de variar el número de neuronas en la capa escondida, se consideró que los mejores resultados se obtuvieron con 13 neuronas en esa capa, con los siguientes parámetros:

- $\varepsilon^{[1]} = 0.4$
- $\varepsilon^{[2]} = 0.4$
- $\alpha^{[1]} = 0.8$
- $\alpha^{[2]} = 0.8$
- Número de ciclos de entrenamiento: 60.000
- Se decrementó el coeficiente de aprendizaje en un 10% cada 10.000 ciclos.

La RN resultante y entrenada generó sus pesos y umbrales por capas en diferentes archivos que contienen: los pesos de la capa de entrada hacia la capa oculta; los pesos de la capa oculta hacia la

capa de salida; los umbrales de la capa de entrada hacia la capa oculta; los umbrales de la capa oculta hacia la capa de salida; información relativa a normalización del histograma hacia la capa de entrada de la red; e información relativa a normalización del histograma hacia la capa de entrada de la red. Esta información define totalmente la RN entrenada, y se transportó para su uso bajo ambiente *Windows* en la fase de reconocimiento.

2.1. Reconocimiento de Mineral por Rectángulo

En el proceso de reconocimiento se implementó la modalidad de selección de área. El usuario define un rectángulo de tamaño variable de la imagen con el *mouse*, y luego comienza el siguiente procedimiento (cuyos algoritmos se muestran en Anexo A):

a. Genera histograma de área seleccionada almacenándola con 256 valores;
b. Transforma *paleta* de 256 valores a un intervalo de 22 y colocando el valor 23 como 0, con el fin de generar el vector de entrada a la RN;
c. Normaliza tamaño de área seleccionada a tamaño estándar con que se entrenó la RN (10x10);
d. El vector debe normalizarse nuevamente debido a que debe estar en un rango (0,1);
e. El vector de entrada a la RN se procesa por intermedio del algoritmo de propagación;
f. Siete (7) neuronas de salida representan un mineral (covelina usa 2 neuronas). Si hay alguna neurona excitada sobre el valor 0.6 (determinado por el experto), entonces es altamente factible que el área seleccionada sea dicho mineral. Si ninguna neurona alcanza el umbral, la muestra no es suficientemente clara, y se excita la octava neurona que identifica el Fondo. La relación es:

oi[0]	Calcopirita
oi[1]	Calcosina
oi[2] oi[3]	Covelina
oi[4]	Bornita
oi[5]	Pirita
oi[6]	Energita
oi[7]	Fondo

2.2. Presencia Porcentual de cada Mineral en la Muestra

Este proceso realiza un recorrido exhaustivo de la imagen, barriéndola en su totalidad por medio de dos ciclos en el cual contempla el tamaño del área seleccionada por el experto (para este caso fue 5x5). El resultado es la cantidad de veces que reconoció el mineral i-ésimo. Luego los resultados en relación a estos valores determinan la relación porcentual de cada mineral en la muestra.

3. Interfaz Desarrollada

La figura 1 muestra la interfaz desarrollada, en donde existe la opción de operar de dos maneras con una imagen cargada: Reconocimiento y Porcentajes.

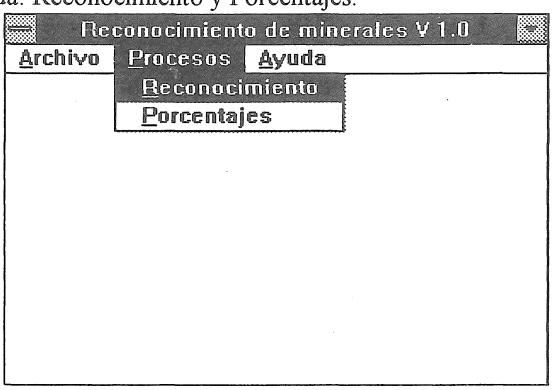


Fig. 1: Interfaz del sistema.

3.1. Modalidad Reconocimiento

Para ejecutar esta opción se debe marcar previamente un cuadrado en el área de trabajo. Una vez elegida un área en la imagen del mineral que se desea reconocer, se elije ésta opción para que el sistema indique de que mineral se trata (Fig. 2). La respuesta se indica en una ventana de diálogo, en la que aparece el nombre del mineral identificado, junto con el grado de certeza del reconocimiento (Fig. 3). Puede darse el caso que se entregue como resultado más de un mineral, caso en el que se agrega la información del grado de confiabilidad de cada uno de ellos. También

puede darse el caso la muestra entregada no sea ninguno de los minerales que es capaz de reconocer esta aplicación, caso en el cual entregará un mensaje de mineral no reconocido.

Fig. 2: Selección de un área a examinar.

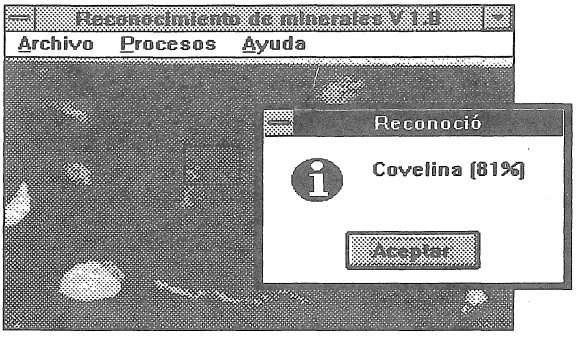


Fig 3: Reconocimiento de un mineral.

3.2 Modalidad Porcentajes

Realiza un proceso de reconocimiento en toda la imagen, entregando como resultado el porcentaje total que ocupa cada mineral en la muestra. (Fig. 4).

Fig. 4: Cálculo porcentual de cada mineral en la muestra.

4. Conclusiones y Consideraciones

Dado que la aplicación opera con imágenes digitalizadas, se debe atender dos puntos importantes:

- Las imágenes con las que se entrenó la RN son de formato TIFF paletizado, por lo que cualquier otro tipo de imagen será incomprensible para la aplicación.
- Dado que las imágenes están en 256 colores, se recomienda utilizar un equipo que soporte *Windows* con esta cantidad de colores o mas. A pesar que la RN opera correctamente al trabajar con 16 colores, se degrada la imagen, haciéndola menos visible, pudiendo ésto acarrear problemas al seleccionar las áreas de reconocimiento.

Los resultados obtenidos con el uso de RN en el reconocimiento de minerales son:

- En la fase de entrenamiento se obtuvo un reconocimiento correcto de 97% identificado.
- En la fase de prueba con imágenes nuevas la identificación correcta fue de 91%.
- Se usaron 160 patrones de entrenamiento y se probaron 120 patrones.

Actualmente, se está en la fase de incorporar una nueva capa escondida, lo que permitirá mejorar eficiente y significativamente los resultados.

Se observó que varios de los problemas de mala clasificación se debe al hecho que algunas regiones de imágenes presentan superposición de 2 o más minerales, lo que hace que los histogramas sean poco "claros". Una ventana de menor tamaño ayuda a resolver este problema.

Otra fuente de error corresponde al proceso de revelado, pelicula utilizada, papel de la fotografía, y proceso de digitalización, entre otros. Estas condiciones de procesamiento tienen gran relevancia en los resultados. Para eliminar las posibles fuentes de error, se está trabajando sobre datos obtenidos directamente desde una cámara de video instalada en el microscopio, desde donde se obtuvieron las fotografías de las muestras de las briquetas de los minerales.

De los resultados obtenidos, y de acuerdo a la opinión de usuarios en el Centro de Investigaciones Minero Metalúrgico (CIMM), consideramos que el tipo de RN descrita en este trabajo, permite automatizar satisfactoriamente el proceso de reconocimiento de minerales.

Reconocimientos

Agradecemos la colaboración C. Aguirre, G. Salini y P. Verdugo financiados por la Comisión Nacional de Ciencia y Tecnología (CONICYT) y el Centro de Investigaciones Minero Metalúrgico (CIMM) a través del proyecto FONDEF IN-03.

Referencias

[1] Gallant, S., *Neural Networks Learning and Expert Systems*, MIT Press, Cambridge, Mass., 93.

[2] Gonzalez, R. and Woods, R., *Digital Image Processing*, Addison-Wesley Publ., 1992.

[3] Rumelhart, D.; McClelland J. and PDP Research Group, *Parallel and Distributed Processing*; Vol.1, MIT Press, Cambridge, Mass., 1986.

ANEXO

Algoritmo a:

```
memset(&r_histo[0],0, 22*sizeof(int));
memset(&g_histo[0],0, 22*sizeof(int));
memset(&b_histo[0],0, 22*sizeof(int));
memset(&rgb_histo[0],0,67*sizeof(int));
for(x=0;x<256;x++)        paleta[x]=0;
for(i=PosYs;i<PosYi;i++)
   for(j=PosXs;j<PosXi;j++) {
            paleta[im[i][j]]++;
            r_histo[aprox(trgb[a][0]/3)]++;
            g_histo[aprox(trgb[a][1]/3)]++;
            b_histo[aprox(trgb[a][2]/3)]++;
   }
```

Algoritmo b:

```
/* Forma patrón de 66 componentes (3x22), uniendo histogramas RGB */
j=0;
for(k=0;k<=21;k++)   rgb_histo[j++]=r_histo[k];
for(k=0;k<=21;k++)   rgb_histo[j++]=g_histo[k];
for(k=0;k<=21;k++)   rgb_histo[j++]=b_histo[k];
/* Generación de Intervalos */
intervalo=0;
sum=0;
j=0;
for(k=0 ;k<=66 ;k++)
{  sum=sum+rgb_histo[k];
   j++;
   if(k==66) j=A_INTER;
   if(j==A_INTER)
   {        j=0;
            integ[intervalo++]=sum;
            sum=0;
   }
}
integ[22]=0;
```

Algoritmo c:

```
std = 15*15;
largox=PosXs-PosXi;
largoy=PosYs-PosYi;
muestra = largox*largoy;
muestra=(muestra>0)?muestra:-muestra;
coef=muestra/std;
for (x=0;x<23;x++) {
```

```
        resto=integ[x]/coef-(int) integ[x]/coef;
        ti[x]=(resto>0.5)?(integ[x]/coef)+1:(integ[x]/coef);
    }
```

Algoritmo d:

```
    for(x=0;x<23;x++)
    if (max[x] > 0)  tic[x]=((rri-ri)*ti[x]+max[x]* ri- min[x]*rri)/(max[x]-min[x]);
    else tic[x]=ti[x];
```

Algoritmo e:

```
    for(x=0;x<11;x++) { /* Propagación Input to Hidden */
        zi[x]= 0;
        for(y=0;y<23;y++)      zi[x] += pih[x][y] * tic[y];
        zi[x] += uih[x];
        hi[x] = 1 / (1 + exp(-zi[x]) );
    }
    for(x=0;x<6;x++) { /* Propagación  Hidden to Output */
        zo[x]= 0;
        for(y=0;y<11;y++)      zo[x]+= pho[x][y]*hi[y];
        zo[x]+= uho[x];
        oi[x] = 1/(1+exp(-zo[x]));
    }
```

Algoritmo Porcentaje:

```
  for(i=0;i<7;i++)    porcentaje[i]=0;
  for(PosXs=0;PosXs<320;PosXs+=INCRE) {
    PosXi=PosXs+INCRE;
    if (PosXi>=320) PosXi=319;
   for(PosYs=0;PosYs<200;PosYs+=INCRE)
   {     PosYi=PosYs+INCRE;
         if (PosYi>=200) PosYi=199;
         calcula_mineral();
         mayor=oi[0];
         mayorind=0;
         for(i=1;i<6;i++)       if (oi[i]>mayor) {
                                        mayor=oi[i];
                                        mayorind=i;
                                }
        if (mayor<.6)   mayorind=6;
        porcentaje[mayorind]++;
        total++;
```

CODIFICACION DE VIDEO DIGITAL BASADO EN LA ESTIMACION IMPLICITA DE MOVIMIENTO INTER-IMAGEN UTILIZANDO REDES DE NEURONAS

Por el Dr. Eduardo J. García García

Profesor Investigador del Departamento de Ciencias Computacionales
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México
Carretera Lago de Guadalupe Km. 3.5
52926 Atizapán de Zaragoza, Edo. de México
e-mail : egarcia@campus.cem.itesm.mx

RESUMEN

En este trabajo se presenta la aplicación de las Redes de Neuronas Artificiales (RNA) en la codificación de video digital. Se aborda en particular la codificación inter-imagen con el objeto de eliminar en lo posible la información visual redundante en el tiempo. Dicha codificación inter-imagen consiste en un proceso de predicción donde se intenta estimar la imagen a codificar, a partir de la imagen precedente compensada en movimiento. En los sistemas normalizados de codificación de video, la predicción inter-imagen es realizada con la ayuda de métodos iterativos de estimación de movimiento como el *Block Matching*. En este artículo se muestra que es posible substituir el proceso de estimación/compensación de movimiento por una RNA. El predictor neuronal es capaz de obtener valores de predicción de las imágenes a codificar basándose en un aprendizaje sobre las imágenes precedentes. Los valores de predicción llevan implícitamente la evolución del movimiento en la escena, de donde se sugiere que la RNA es una forma de predictor con una estimación implícita de movimiento.
Palabras Clave : redes de neuronas artificiales, algoritmo de retropropagación, block matching, codificación inter-imagen, estimación de movimiento, compensación de movimiento.

I. Introducción

El desarrollo vertiginoso de la tecnología nos conduce poco a poco a abandonar los sistemas analógicos y remplazarlos por sistemas digitales. En este contexto, el procesamiento y la transmisión del video forman parte de esta evolución. Sin embargo, la representación digital del video nos confronta al problema de manipulación de grandes volúmenes de información. Es por esto que surge la necesidad de buscar métodos y algoritmos capaces de codificar y comprimir el video digital, sin introducir degradación en la información visual. Respondiendo a esta necesidad, los organismos internacionales de normalización proponen diversos esquemas de compresión de video digital. Dichos esquemas normalizados involucran a su vez dos procedimientos de codificación : la codificación espacial o *intra-imagen* y la codificación temporal, llamada también *inter-imagen*.

La codificación *intra-imagen* consiste en la eliminación de la información redundante en el espacio, proyectando la señal imagen sobre una base ortogonal. Dicha proyección permite pasar del dominio imagen al dominio de las frecuencias espaciales, donde se realiza la cuantificación de aquellas no reelevantes, desde el punto de vista de la percepción humana.
La codificación *inter-imagen*, en cambio, permite la eliminación de la información redundante en el tiempo mediante la predicción de imágenes. Dicha predicción consiste en generar una imagen muy cercana a la que se desea codificar, utilizando las imágenes precedentes y la estimación de movimiento de los objetos en la escena. De este modo, la transmisión o almacenamiento de la diferencia entre la imagen original y su predicción permite un ahorro importante en el volumen de información del video.
La Fig. 1 muestra la estructura de base de un codificador híbrido de video. Las diferencias básicas entre cada norma de codificación propuestas por los organismos internacionales residen en la elección de la cuantificación de frecuencias espaciales, el método de estimación de movimiento, la codificación entrópica de canal y la regulación del flujo de datos a la salida del codificador. Sin embargo, el esquema en todo caso es similar al de la Fig. 1.

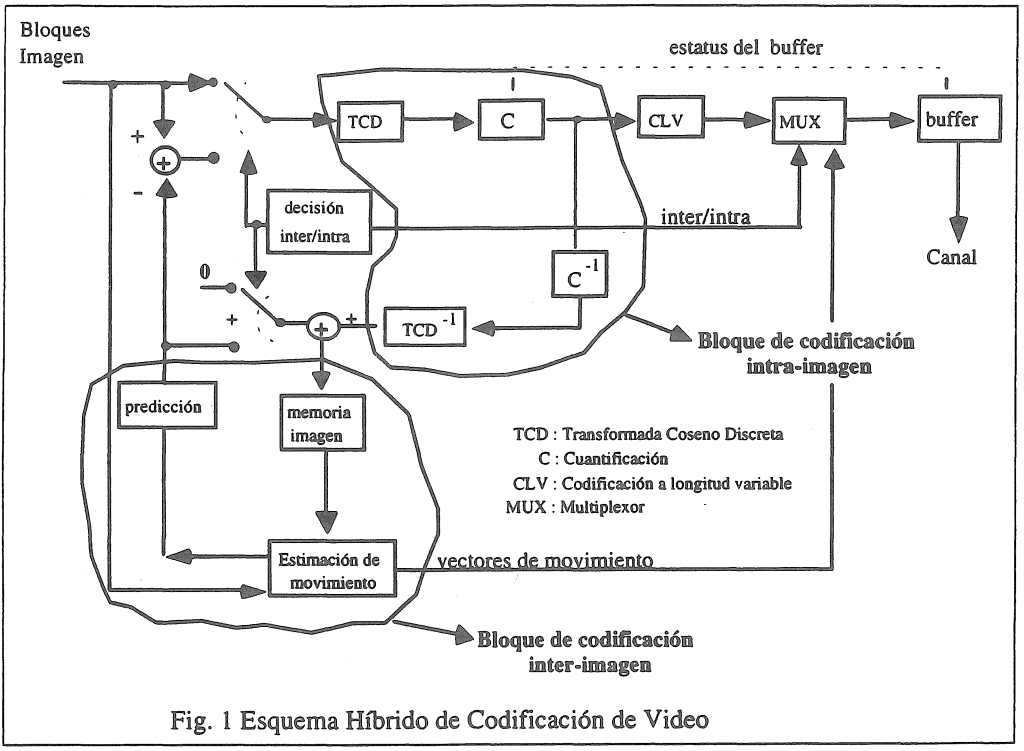


Fig. 1 Esquema Híbrido de Codificación de Video

En el presente artículo se abordará el problema de la codificación inter-imagen, la cual es decisiva en la adecuada compresión del video digital.

Primeramente, se hará una descripción breve del esquema de codificación utilizado en nuestra investigación. Se eligió trabajar sobre un esquema del European Telecommunications Standards Institute (*ETSI*)[1] para transmisión de video sobre canales inferiores a 17 Mbits/s . A continuación se detallará el método de codificación inter-imagen utilizado en dicho esquema, haciendo énfasis especial en la estimación de movimiento. Enseguida, explicaremos el principio de predicción propuesto y su integración al esquema de codificación. Finalmente, mostraremos los resultados obtenidos en la compresión de video digital formato TV según la Rec. 601 del Comité Consultivo Internacional de Radiocomunicaciones (CCIR)[2].

II. El Esquema de Codificación Video ETSI

En el marco de este estudio, se escogió un esquema de codificación video muy cercano a la *Recomendacion H.261*[3] de *CCITT* para video-conferencia : el esquema *ETSI* definido en el proyecto Europeo *RACE 1018*[1]. A diferencia de la *Rec. H.261*, usada en la transmisión de video-conferencia entre 64 y 384 Kbps, el esquema *ETSI* permite la transmisión de imágenes en formato televisión sobre canales de a lo más 17 Mbits/s. Este esquema es derivado de la norma de transmisión de televisión digital con calidad estudio a 34 Mbits/s[1].

Algunas de las características del esquema *ETSI* son las siguientes :

- Codificación de imágenes entrelazadas en formato *CCIR Rec. 601*[2]. Codificación trama por trama.
- Tres modos de codificación son posibles : codificación intra-imagen, codificación inter-imagen y codificación inter-imagen con compensación de movimiento. La elección del tipo de codificación se realiza sobre cada macro-bloque de 16x8 pixels.
- Estimación de movimiento usando el método de *Block Matching*
- Predicción a partir de la imagen precedente por medio de interpolación lineal, en caso de movimientos en fracción de pixel.
- Compensación de movimiento sobre macro-bloques de 16x8 pixels.
- Codificación por transformación : Uso de la *Transformación Coseno Discreta* (TCD) sobre bloques imagen 8x8 pixels. Dicha transformación se aplica sea sobre bloques de la imagen directamente, sea sobre un bloque de diferencias. Este último se obtiene de sustraer a un bloque de imagen original, un bloque constituído de valores de predicción (inter-imagen).
- Codificación entrópica de los coeficientes TCD resultantes.

La eficiencia de este esquema de codificación depende estrechamente de los módulos de estimación/compensación de movimiento y de la cuantificación de los coeficientes TCD. El esquema de codificación ETSI sigue la estructura mostrada en la Fig. 1.

III. La Estimación de Movimiento usando Block Matching

La estimación de movimiento es una técnica que nos permite evaluar el movimiento de los objetos en la escena. En el codificador, esta información es utilizada para generar imágenes de predicción, que siendo suficientemente acertadas, permitirán disminuir el volumen de información mediante la diferencia de cada imagen original y su predicción. Cabe notar que el decodificador del video debe estar en capacidad de reconstruir la predicción utilizada, para lo cual, necesita la información relativa al movimiento de los objetos en la escena. Esto provoca la transmisión o almacenamiento de información suplementaria.

Uno de los métodos de estimación de movimiento más comunes en los esquemas de codificación es el denominado "apareo de bloques" o *Block Matching*. El *block matching* consiste en la búsqueda iterativa de un conjunto de pixels (macro-bloque) en la imagen *t*, dentro de una ventana de pixels en la imagen *t-1*. Si la búsqueda da resultados positivos, el macro-bloque puede ser representado por su homónimo de la imagen precedente afectado por un vector de movimiento $(x,y)^T$, como se ve en la Fig. 2.

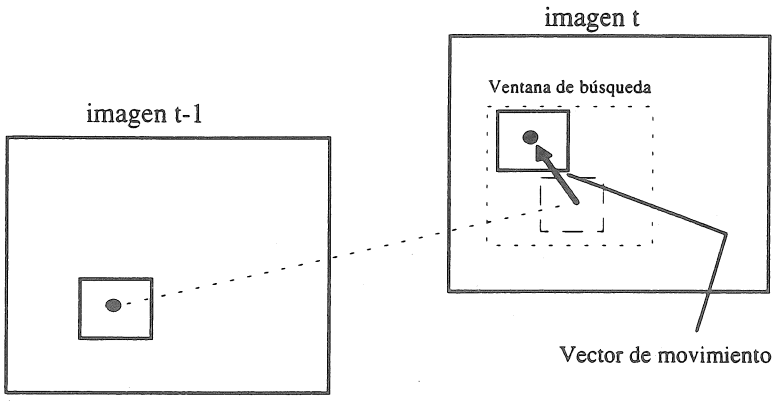


Fig. 2. El método de Block Matching

En el caso del esquema de codificación ETSI, la búsqueda del macro-bloque es de tipo *full-search*, sobre una ventana de ±16 pixels horizontalmente por ±8 pixels verticalmente. El criterio de "apareo" (match) utilizado es la minimización del error cuadrático medio (Mean Square Error) :

$$\mathbf{MSE}\ (x,y)^T = \frac{1}{(16)(8)} \sum_{(i,j)\ \in\ \text{ventana}} [p(i,j,k) - p(i+x, j+y, k+1)]^2$$

siendo $p(i,j,k)$ el valor de un pixel en la posición (i,j) de la imagen k y $(x,y)^T$ el vector de movimiento en consideración.

Como se podrá constatar, el hecho de efectuar una búsqueda exhaustiva del bloque involucra un número considerable de cálculos de MSE. Así pues, es necesario encontrar métodos menos costosos, en términos de consumo de tiempo y complejidad. Varias técnicas de búsqueda son propuestas en la literatura [4], pero los resultados no son del todo satisfactorios.

Una vez calculado el movimiento de los macro-bloques es posible, como ya se dijo, representarlos por sus homónimos en las imagenes precedentes pero con el desplazamiento indicado por los vectores de movimiento respectivos. A este proceso se le conoce como compensación de movimiento. De este modo, los bloques compensados en movimiento son utilizados como una predicción de los bloques originales en la imagen t.

IV. La Estimación Implícita de Movimiento Usando RNA

Como ya se mencionó, la estimación de movimiento usando la técnica de *block matching*, es una solución bien definida y adoptada en la mayor parte de los esquemas de codificación de video. Sin embargo, la complejidad del algoritmo depende estrechamente del tamaño de la ventana de búsqueda así como del criterio de semejanza, en este caso el MSE.

En este contexto, se efectuó un estudio que permite mostrar la factibilidad y el interés de utilizar RNA para realizar la predicción de imágenes, sin necesidad de una estimación/compensación explícita de movimiento.

Las RNA han sido ya utilizadas en aplicaciones, en las cuales, se busca predecir comportamientos a partir de una serie de observaciones en el tiempo. Tal es el caso de predicciones en los índices bursátiles, predicción en el consumo de agua, luz, etc. En nuestra aplicación, las RNA serán utilizadas como un predictor de pixels de una imagen. El conjunto de valores de entrada del predictor es la vecindad causal de cada pixel a predecir. Primeramente, la predicción se realizará pixel por pixel considerando sólo el entorno inmediato en la misma imagen y aquel de la imagen precedente.

El modelo de RNA escogido es el perceptron multi-capas, en el cual los pesos de conexión serán calculados mediante el algoritmo de retropropagación de gradiente con factor de modulación adaptativo[5].

Las entradas del perceptron multi-capas son constituídas por los pixels vecinos del valor a predecir, tanto en el dominio espacial (misma imagen), tanto como en el dominio temporal (imagen precedente).

Esto es debido a que el valor de un pixel *p* esta estrechamente ligado a los valores de sus vecinos (correlación espacial) y es probable que el pixel *p* se encuentre en la misma región de la imagen precedente (correlación temporal). Así pues, es posible distinguir dos ventanas causales como se muestra en la Fig. 3.

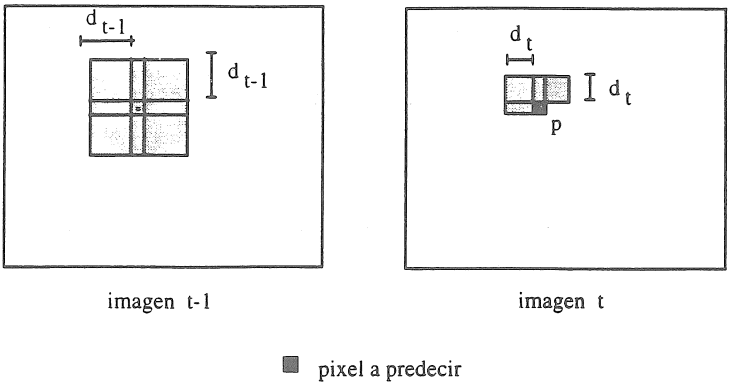


Fig. 3. Ventanas consideradas en la predicción de un pixel

Podemos ver que el número de pixels considerado para la predicción depende de los valores d_{t-1} y d_t que definen las ventanas causales en la imagen *t-1* y la imagen *t* respectivamente. Se escogieron los valores $d_{t-1}=4$ y $d_t=2$ considerando una distancia de movimiento inferior a 1/2 bloque de tamaño 8x8.

De este modo, el número de pixels considerados en la predicción es :

$$\{4(d_{t-1}^2 + d_{t-1}) + 1\} + 2(d_t^2 + d_t) = 81 + 12 = 93$$

Los experimentos llevados a cabo sobre secuencias de video de prueba, sugeridas por los organismos de normalización, mostraron que los valores escogidos para d_{t-1} y d_t son adecuados. La consideración de ventanas de mayor tamaño no introduce una mejora sustancial en la predicción. En cambio, se aumenta el costo en términos de complejidad de cálculo y aprendizaje de la RNA. Más aún, los estudios de movimiento efectuados por otras instancias mostraron que es poco frecuente encontrar desplazamientos superiores a ±4 pixels, en el caso de secuencias de video formato televisión[6].

En cuanto a la RNA utilizada, se trata de un perceptron multicapas con 93 neuronas en la capa de entrada, correspondientes a las vecindades causales, y una neurona en la capa de salida correspondiente a la predicción realizada. La estructura de la RNA se muestra en la Fig. 4.

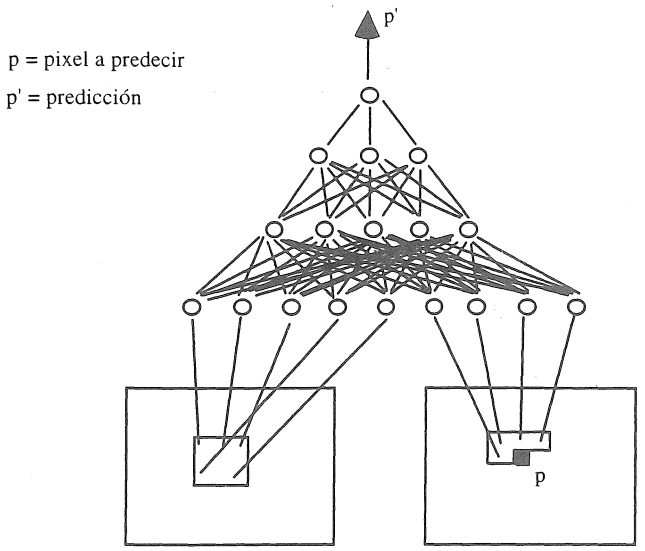
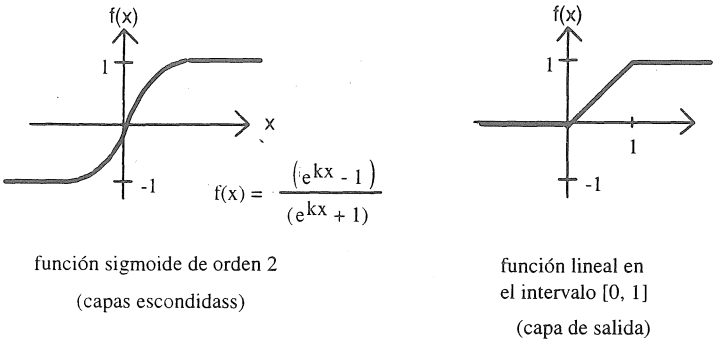


Fig. 4. Predicción utilizando una RNA multi-capas

Sin embargo, el número de capas escondidas así como el número de neuronas en cada una de ellas es menos evidente a obtener. Hasta ahora, no existe un método eficiente para determinar el número de neuronas en las capas escondidas de una RNA multi-capas, en función de sus entradas y salidas. Se probaron múltiples configuraciones, todas ellas de forma a generar una estructura en pirámide, es decir, cada capa escondida contiene un número menor de neuronas a medida que se encuentra más cerca de la capa de salida. Esto se basa en el hecho que en cada capa se efectua una proyección de las entradas sobre un espacio de menor dimensión, lo cual nos lleva a una síntesis de la información, que es lo que se busca. La elección de capas ocultas de mayor tamaño a cada vez, ocasiona una expansión del problema en espacios de mayor dimensión, lo que implica una descomposición en vez de una síntesis[**6**].

En cuanto a la función de activación de las neuronas, se escogió la tangente hiperbólica (***tanh***). Esta función permite un rango de valores de salida de las neuronas en el intervalo [-1, +1], como se ve en la Fig. 5. La función ***tanh*** es idéntica a la función sigmoide de orden 2 y es utilizada al interior de la RNA, es decir en las capas ocultas. El objetivo es generar una amplia dinámica en los valores de salida de dichas neuronas. Sólo la neurona de salida tiene una función lineal, para limitar su salida en el intervalo [0, 1].



función sigmoide de orden 2
(capas escondidass)

función lineal en
el intervalo [0, 1]
(capa de salida)

Fig. 5. Funciones de activación del perceptron multi-capas

Los valores de los pixels en la entrada del perceptron pueden tomar valores entre 1 y 255. En este caso, el uso directo de estos valores ocasionaría la saturación de la función de activación de las neuronas y ninguna solución será posible. Por este motivo, los valores en entrada son normalizados dividiéndolos entre 255. Con esto se logra obtener valores en el intervalo [0,1]. En cuanto a la salida de la RNA, los valores obtenidos se encuentran también en el rango de 0 a 1, por lo que la multiplicación por 255 permite encontrar un valor válido de predicción en el dominio imagen.

Finalmente, y como ya se mencionó, el algoritmo de aprendizaje de la RNA es la retropropagación de gradiente con factor de modulación adaptativo.

V. Resultados Obtenidos

La Tabla 1 presenta los resultados obtenidos para la secuencia de video *mobile_calendar*, utilizada en las pruebas de codificadores normalizados (cortesía del *Centro Común de Estudios en Teledifusión y Telecomunicaciones* de Francia). Esta secuencia consiste en un muro con tapiz multicolor en donde se desplaza un tren a escala de derecha a izquierda. Al mismo tiempo, un calendario sobre el muro se desplaza de arriba a abajo. La complejidad de la escena así como los movimientos múltiples la hacen difícil de procesar.

Se compararon los resultados con la salida del predictor original usado en el esquema *ETSI*, que involucra la estimación de movimiento por *block matching*. La secuencia de entrenamiento usada para el aprendizaje de la RNA consistió en las 2 primeras imágenes de la secuencia *mobile_calendar*. Se eligieron un total de 6000 muestras uniformemente espaciadas.

Los mejores resultados se obtuvieron con una RNA de 5 capas : 93 en la capa de entrada, 3 capas escondidas con 50, 25 y 10 neuronas respectivamente y una capa de salida de una neurona.

Método Usado	√EQM
Perceptron (93-40-1)	87.2
Perceptron (93-60-1)	85.4
Perceptron (93-50-25-10-1)	14.3
Block Matching	9.6

Tabla 1. Comparación de diferentes predictores para mobile_calendar
(Error de predicción de la imagen 2 de mobile_calendar)

Podemos notar que el hecho de aumentar el número de capas escondidas permite obtener mejores resultados. Sin embargo, este aumento genera también un aumento considerable en el tiempo de aprendizaje. Por supuesto, la implementación del algoritmo en una máquina paralela permitiría eliminar este problema. Cabe destacar que una vez entrenada la RNA, su funcionamiento como predictor es mucho más rápido que la búsqueda exhaustiva del block matching.

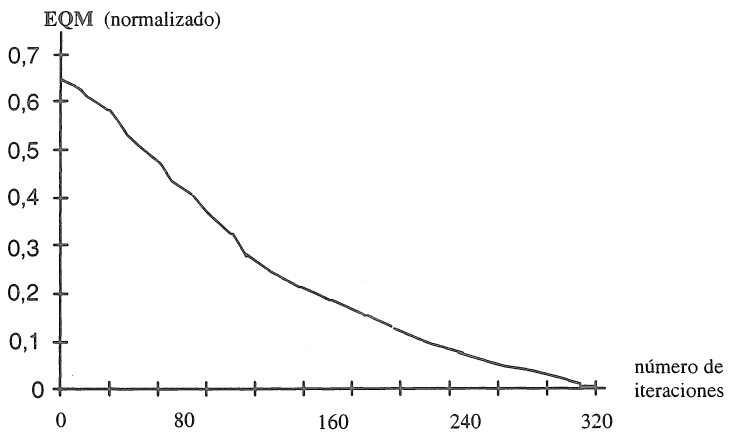


Fig. 6. Convergencia del algoritmo de retropropagación para la RNA de 5 capas.

Se puede observar en la Fig. 7 que los resultados, en términos de error cuadrático medio, son muy cercanos. Sin embargo, desde el punto de vista visual, la predicción por RNA nos da imágenes más claras y agradables.

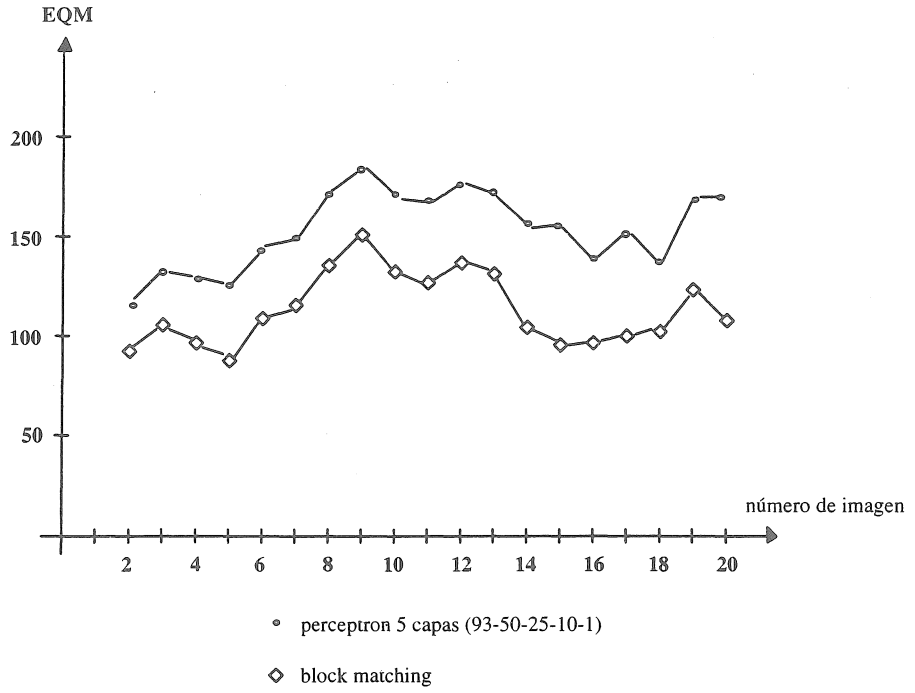


Fig.7. Comparación de los errores de predición sobre la secuencia mobile_calendar

Puesto que el movimiento de los objetos en la escena se encuentra implícito en la predicción dada por la red, no es necesario transmitir o almacenar ningún tipo de información adicional, como es el caso de los vectores de movimiento en el esquema convencional. Sin embargo, si existe la necesidad de hacer llegar los pesos de conexión de la RNA al decodificador. Así pues, la información relativa al movimiento, en el caso del esquema clásico, es remplazada por los pesos de conexión lo cual no afecta el volumen global de información video codificada. De cualquier forma, la simplicidad del predictor neuronal y su rapidez, comparado con el método de *block matching*, lo coloca todavía como una buena elección.

Finalmente, se integró el predictor neuronal en el esquema de codificación ETSI. En este caso, la parte de predicción y estimación de movimiento fue remplazada por el predictor neuronal. Los primeros experimentos se realizaron sobre el esquema de codificación en lazo abierto, es decir, los valores de la vecindad causal del pixel a predecir son tomados de las imágenes originales. Esto nos permitió evaluar el comportamiento del sistema sin el efecto de deriva. Hay que recordar que la deriva consiste en la acumulación del error de codificación debido al uso de informaciones codificadas precedentemente en vez de información original. Es esta deriva la que justifica el refrescado periódico de la secuencia de video.

Por otra parte, con el fin de eliminar problemas en la predicción de los pixels de los bordes de las imágenes, el entrenamiento de la RNA se efectuó tomando un marco de 4 pixels alrededor de cada imagen con valor igual al promedio de luminosidad. Esta elección se basa en el hecho que el sistema visual humano es más o menos sensible a las variaciones alrededor de la media de luminosidad de la escena que percibe.

El flujo de datos escogido para las simulaciones fue de 11 Mbits/s. Sólo la componente de luminosidad (imagen blanco/negro) fue considerada. Se codificaron las primeras 20 imágenes de la secuencia *mobile_calendar*. Se midió la relación señal/ruido (Signal to Noise Ratio SNR) y se comparó con el esquema clásico con estimación de movimiento por block matching. Se observó una ganancia de al menos 2% desde el punto de vista SNR. Se observó también que la calidad visual de las imágenes reconstruídas así como la rapidez de codificación son superiores al del esquema convencional *ETSI*. Los resultados se observan en la Fig. 8.

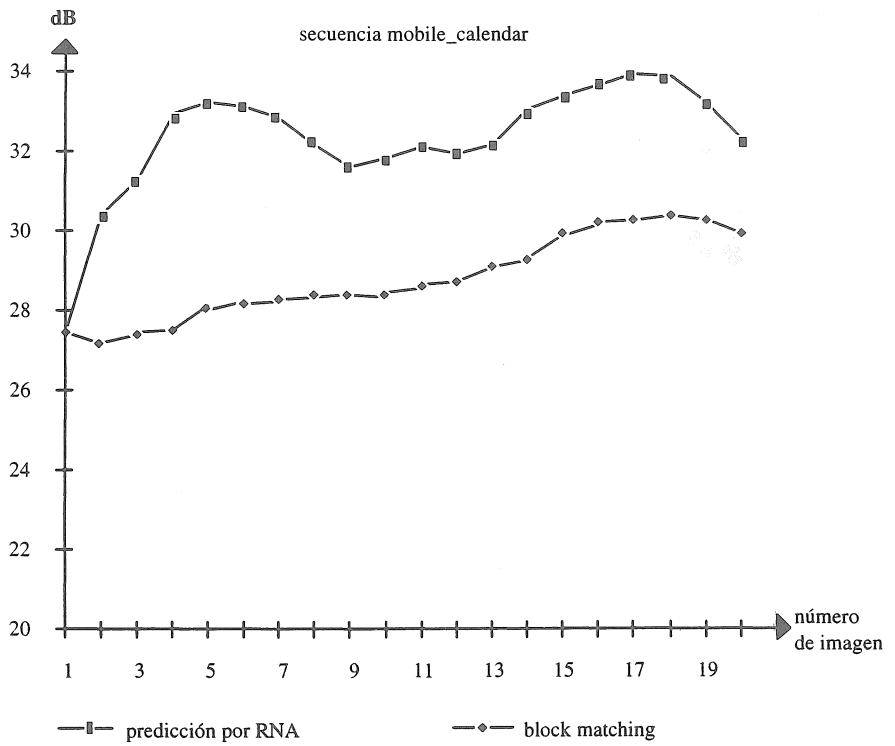


Fig. 8. Comparación de predictores en el esquema ETSI

VI. Conclusiones y Perspectivas

Hasta aquí, podemos concluir que el uso de RNA para la predicción de señal imagen es factible y se obtienen resultados, en cuanto a la calidad visual de las imágenes, mejores a los métodos convencionales. Dejando aparte el problema del aprendizaje, el uso de la RNA tiene sus ventajas en cuanto a la rapidez de ejecución y la simplicidad de su implementación. La RNA asegura un buen almacenamiento de las variaciones espacio-temporales de las imágenes.

Los experimentos realizados sobre un esquema de codificación real confirman la eficiencia del predictor neuronal, aunque hasta ahora sin introducir deriva en el sistema (lazo abierto). Nos queda verificar el comportamiento del predictor neuronal en lazo cerrado. En este caso y dada la estructura de codificación por bloque del esquema ETSI, tendremos que pensar en implementar un predictor neuronal que involucre varios perceptrones en paralelo. Esto se debe a que en algunos casos, no contaremos con la vecindad causal. Esto se ejemplifica en la Fig 9. Sin embargo, el nuevo paradigma neuronal parece prometedor y se espera obtener resultados satisfactorios a corto plazo.

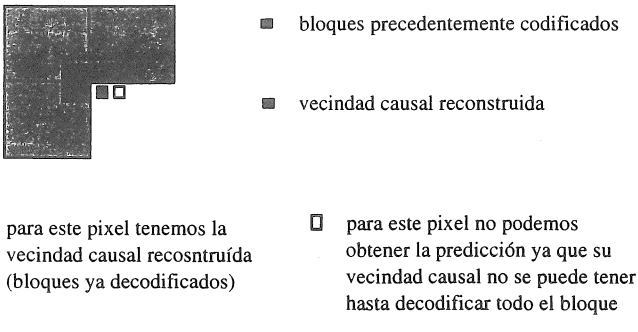


Fig. 9. Problema de vecindad causal en lazo cerrado.

BIBLIOGRAFIA

[1] ETSI, "Network Aspects Digital Coding of Component TV Signals for Contribution Quality Applications in the range 34-45 Mbits/s", Draft prETS 300 174, June 1991

[2] CCIR, "Recommendation 601-I Digital Methods of Transmittig TV Information", 1986

[3] CCITT, "Recommendation H.261, Video codec for audiovisual services at px64 kbps", 1990

[4] Choquet B., "Estimation de Mouvement et Segmentation Spatio Temporelle en Sequences d'images", Tesis de Doctorado, Univ. Rennes I, Sep. 1988

[5] Qiu G. et al, "Accelerated Training of Backpropagation Networks using Adaptive Momentum Step", Electronics Letters, Vol. 28, N. 4, Feb. 1992.

[6] García E., "Etude et Utilisation des Réseaux de Neurones pour le Codage d'images de TV", Tesis de Doctorado, Univ. Rennes I, Dic. 1994.

MANEJO DE ARCHIVOS REPLICADOS BAJO TRANSACCIONES DISTRIBUIDAS EN EL SISTEMA DGDBM

Orlando CASTRO
María Consuelo FRANKY

Depto. de Ingeniería de Sistemas y Computación
Universidad de los Andes
Apartado Aéreo 4976
Santafé de Bogotá - COLOMBIA
(e-mail: cfranky@uniandes.edu.co)

RESUMEN:

Este artículo describe el diseño y la implantación de los protocolos de replicación de archivos denominados Lectura y Actualización Ponderada (LAVP) y Copia Primaria Copias Secundarias (CPCS) para el sistema DGDBM. El sistema DGDBM es un API que ofrece funciones para el desarrollo de aplicaciones distribuidas en una red de computadores, con ejecución de conjuntos de operaciones en forma transaccional y manipulación de archivos independientemente de su ubicación por parte del programador de aplicaciones. En el artículo se analizan algunas alternativas para implantar protocolos de replicación de archivos, conservando la transparencia a la distribución en los servicios ofrecidos al programador. Además se describen las características y la arquitectura de la implantación final de estos protocolos en el sistema DGDBM.

PALABRAS CLAVE:
Sistemas Distribuidos, Sistemas de Archivos Distribuidos, Sistemas de Transacciones Distribuidas, Protocolos de Replicación de Datos.

1. INTRODUCCIÓN

El uso extendido y cada vez más frecuente de redes de computadores genera la necesidad de herramientas que faciliten la programación de aplicaciones y que exploten las ventajas de contar con una mayor cantidad de dispositivos de almacenamiento y un mayor número de procesadores. Estas herramientas deben buscar aumentar la probabilidad de trabajar con datos locales en lugar de remotos, aumentar la disponibilidad de los datos y asegurar la recuperación en caso de fallas mediante copias de respaldo. Además, se requiere que estas herramientas ofrezcan a los usuarios seguridad respecto a la consistencia y corrección de los datos manejados.

DGDBM es una librería de funciones C que el programador puede incluir en sus programas e invocarlas de manera que la distribución de sus archivos en la red sea transparente para él, y en consecuencia le permita desarrollar su aplicación como si estuviera trabajando con datos centralizados <Franky 95>. DGDBM también ofrece el manejo de transacciones distribuidas asegurando su atomicidad y el control de concurrencia. Adicionalmente, DGDBM busca brindar una mayor disponibilidad de los datos mediante el manejo de archivos replicados.

En este artículo se describe el sistema DGDBM y los protocolos de manejo de replicación de archivos que se implantaron en dicho sistema.

2. DESCRIPCIÓN DEL SISTEMA DGDBM

DGDBM utiliza localmente en cada sitio los servicios del manejador de archivos GDBM <Nelson,93>. Este manejador permite la manipulación de archivos mediante acceso directo utilizando técnicas de hashing extendido. Cada registro de un archivo está formado por dos cadenas de caracteres de longitud variable: una llave y un contenido.

DGDBM ofrece los siguientes servicios <Franky, 95>:

- Comienzo y terminación de transacciones distribuidas : el programador dispone de dos funciones para **marcar** el comienzo y terminación de sus transacciones. Todas las operaciones incluidas entre estas marcas estarán sometidas al control transaccional y por lo tanto, contarán con las propiedades de atomicidad, durabilidad, seriabilidad y aislamiento <Ceri 84> .

- Apertura, Cierre y Eliminación de Archivos: son funciones que permiten la manipulación de archivos completos. De estas funciones solo la de Apertura da la **opción** de especificar el sitio en que se desea crear el archivo, cuando se abre un archivo nuevo; las demás funciones utilizan un catálogo de archivos del sistema para establecer el sitio de ubicación de los mismos.

- Consulta, Actualización y Eliminación de Registros: son funciones que permiten el manejo directo de registros de los archivos. Mediante el manejo de candados jerárquicos a nivel de archivo y de registro <Gray 75>, se puede permitir el acceso de varias transacciones en forma concurrente sobre el mismo archivo y/o registro.

- Consulta Secuencial de Registros: son funciones que permiten la consulta secuencial de los registros de un archivo.

En la figura 1. se muestra la arquitectura de un sistema DGDBM, sus agentes y la relación entre ellos.

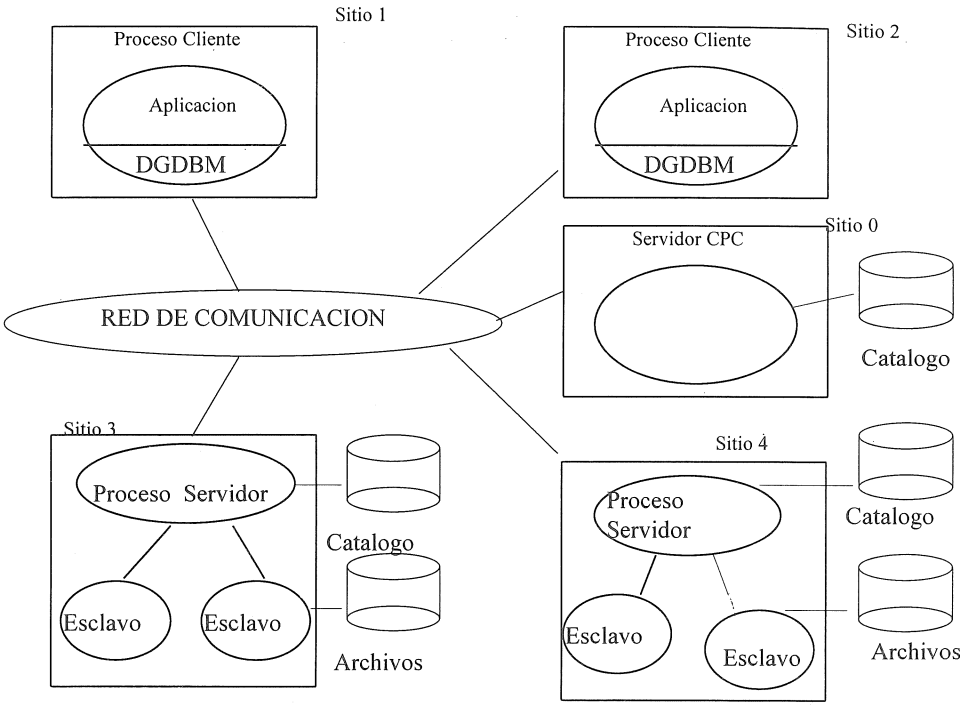


Figura 1. Arquitectura del sistema DGDBM

Una aplicación que utiliza DGDBM se convierte en un **Agente Cliente** en el sistema. Este agente cliente está constituido por el código propio de la aplicación más las funciones del API DGDBM incluido.

En cada sitio componente del sistema se crea, al arrancar el sistema, un **Agente Servidor** encargado de atender las solicitudes de los agentes clientes. Con el fin de mejorar el desempeño del sistema, los agentes servidores generan **Agentes Esclavos** que realizan físicamente las solicitudes de un cliente sobre los archivos, es decir, por cada cliente que esté atendiendo un agente servidor, existe un agente esclavo.

Se maneja un **Catalogo de Archivos**, el cual está replicado en todos los sitios mediante un esquema de copia primaria copias secundarias que es administrado por un **Agente Servidor de la Copia Primaria del Catálogo** <Rangel 94>.

3. PROTOCOLOS DE REPLICACIÓN DE DATOS

Bastantes son los protocolos de replicación que se encuentran descritos y analizados en la literatura. Sin embargo, la mayoría de ellos puede clasificarse en un de los siguientes grupos o familias <Abásolo 94>, <Ahamad 90> <Castro 95>:

- Protocolos de Consenso
- Protocolos de Copias Disponibles
- Protocolos de Copia Primaria-Copias Secundaria

Las características que permiten clasificar un protocolo en uno de estos grupos tienen que ver con la forma en que ofrecen consistencia mutua y corrección de los datos (seriabilidad de copia única, <Mullender 89>).

Los protocolos de consenso establecen un protocolo que exige la participación de varias réplicas del dato, entre ellas al menos una actualizada, para poder realizar una operación. Los de copias disponibles trabajan con las copias que estén activas, pero no siempre garantizan a los usuarios la consistencia de los datos. El protocolo de Copia Primaria-copias secundarias exige la actualización de una misma réplica y permite la consulta de todas; en este caso el usuario decide si la consulta debe hacerse sobre datos actualizados en la copia primaria, o sobre datos no necesariamente actualizados en una de las copias secundarias.

En el sistema DGDBM se implantaron dos protocolos de replicación de datos: Lectura y Actualización por Votación Ponderada y el de Copia Primaria-Copias Secundarias.

3.1 El Protocolo de Lectura y Actualización por Votación Ponderada (LAVP)

Este es un protocolo de consenso en el que a cada réplica del dato se le asigna un número de votos **Vi**, existiendo un total de votos **T** para el sistema, un quórum de lectura **R**, y un quórum de escritura **W**. Además debe cumplirse que: $W > T/2$ y $W + R > T$. Estas condiciones permiten garantizar que no hay dos operaciones simultáneas de escritura sobre un dato, y que una lectura y una modificación tampoco se pueden dar simultáneamente sobre el mismo dato. <Ceri 84>.

Cada vez que se quiera ejecutar una operación sobre un dato replicado debe lograrse la participación de tantas réplicas como se requieran para cumplir con las siguientes condiciones:

- si la operación es una consulta, la suma de los votos de las réplicas participantes debe ser mayor o igual al quórum de lectura.
- si la operación es de modificación, la suma de los votos de las réplicas participantes debe ser mayor o igual al quórum de escritura.

3.2 El Protocolo de Copia Primaria-Copias Secundarias (CPCS)

En este protocolo todas la operaciones de modificación se realizan siempre sobre la misma réplica: la copia primaria, y luego son transmitidas en forma diferida (asincrónica) a las demás réplicas: las copias secundarias. Las operaciones de consulta por otra parte, se pueden realizar sobre la copia primaria, o sobre una cualquiera de las copias secundarias las cuales pueden o no estar actualizadas.

Para garantizar la consistencia de los datos, aún en la ocurrencia de transacciones simultáneas sobre un mismo dato, se debe asegurar que las transacciones que accedan copias secundarias lo hagan en modo de consulta y que todas las transacciones de actualización utilicen la copia primaria. Se establecer así sobre la copia primaria un control de concurrencia similar al que se realiza en datos con copia única.

4. CARACTERÍSTICAS DESEABLES DEL MANEJO DE REPLICACIÓN

- Además de ofrecer consistencia mutua entre las réplicas y corrección de los datos replicados, la replicación debe tener otras características que la hagan más práctica y que permitan aprovechar el máximo de sus ventajas. Entre otras se pueden mencionar las siguientes que fueron implantadas en el sistema DGDBM:
- Transparencia al Usuario: El usuario de los datos (i.e. el programador de aplicaciones en DGDBM), no tiene por qué entenderse con aspectos relacionados con la replicación. Es decir, en la aplicación no debe aparecer código para control y manipulación de la replicación que nada tiene que ver con su propósito original.

- Coexistencia de Varios Protocolos: Las ventajas de la replicación están ligadas en gran parte a la forma en que se usen los datos, siendo más apropiado un protocolo que otro según el caso. Como los datos involucrados en una aplicación pueden ser usados en distintas formas, es conveniente que se cuente con más de un protocolo de replicación y además que puedan manejarse indiscriminadamente datos no replicados, con datos replicados bajo uno u otro protocolo.

- Administración Independiente de la Replicación: La replicación de datos exige administración, es decir, requiere de mecanismos que permitan eliminar una réplica, aumentar el número de réplicas, cambiar el número de votos asignados, el quórum de lectura o escritura, cambiar el sitio de la copia primaria, etc. Estas labores no pueden dejarse a la aplicación que usa los datos, pero son indispensables para el uso práctico de los protocolos de replicación.

5. ALTERNATIVAS DE DISEÑO DE LOS PROTOCOLOS DE REPLICACIÓN

El control de la replicación de archivos puede implantarse de dos maneras: como parte del mecanismo del control transaccional, o como un nivel inferior, independiente del control transaccional sin perjuicio de las propiedades de las transacciones. A continuación se describen las dos alternativas, sus ventajas y desventajas:

5.1 Control de Replicación como Parte del Control Transaccional

En DGDBM el control transaccional lo efectúa el agente cliente, quien debe registrar los sitios participantes y los cambios de estado de la transacción que originó.

En la figura 2, se muestran dos archivos, el archivo A replicado en los sitios 1, 2 y 3, y el archivo B replicado en los sitios 3, 4 y 5. Además, se muestra un cliente C que ejecuta una transacción sobre estos dos archivos.

Como se puede apreciar, el cliente debe establecer comunicación con los servidores de todos los sitios donde haya copias de los archivos involucrados, y que por la lógica del protocolo, deban participar en la operación. Además, si el protocolo de replicación utilizado es LAVP, debe recibir los votos de cada sitio, establecer el momento en que se alcanza el quórum, etc.

Durante el transcurso de la transacción, las operaciones que involucren estos archivos se deben enviar a todos los sitios con réplicas, y al terminar, los cambios de estado (inicial, prevalidar, validar, o anular) debe hacerse en cada sitio de réplica.

Las ventajas de esta forma de controlar la replicación radican en que el mismo control transaccional se puede utilizar para garantizar la consistencia mutua y la corrección de las réplicas, puesto que si se anula la transacción, o se termina con éxito, todas las réplicas necesarias del archivo quedarán en el estado apropiado. Sin embargo, este esquema aumenta la probabilidad de fracaso de una transacción, puesto que exige la disponibilidad de la mayoría (quórum) de los sitios de réplica involucrados, hasta su terminación. Por ejemplo, si una transacción está en estado inicial y uno de los sitios deja de estar disponible, la transacción debe anularse.

De igual manera el tiempo de respuesta al usuario final se aumenta, puesto que las operaciones no pueden considerarse terminadas hasta completar el intercambio de mensajes con todos los servidores involucrados.

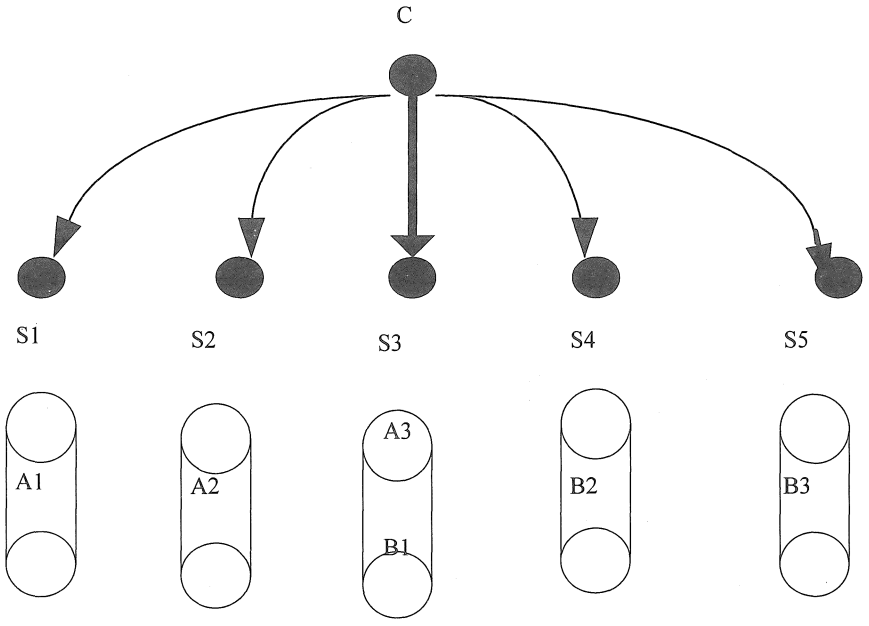


Figura 2. Control del la Replicación dentro del Control Transaccional

5.2 Control de Replicación Independiente del Control Transaccional

El ejemplo que se muestra en la figura 3 corresponde al mismo mostrado en la figura 2, pero como se puede apreciar, el agente cliente que controla la transacción, no se percata de la existencia de replicación de los archivos y tampoco considera la actualización de estas réplicas dentro de la lógica de la transacción. Este agente trata todos los archivos, replicados o no replicados, de la misma forma.

En cambio, aparece un nuevo rol para el agente servidor de un sitio: para cada archivo existe un agente servidor que coordina la actualización de sus réplicas (servidores sitio 1 y sitio 5, en el ejemplo). Este

agente coordinador es el que interactúa con el cliente para todo lo que tenga que ver con el archivo, y, una vez terminada la transacción, es el encargado de actualizar las réplicas y garantizar la consistencia mutua y la corrección de las copias (mediante la lógica del protocolo respectivo).

De esta forma no se requiere la disponibilidad de la mayoría de los sitios de réplicas hasta el final de la transacción sino solo la del coordinador de cada archivo, lo que aumenta la probabilidad de éxito de la transacción y permite, además, disminuir el tiempo de respuesta al usuario final, puesto que disminuye el número de mensajes intercambiados con el cliente.

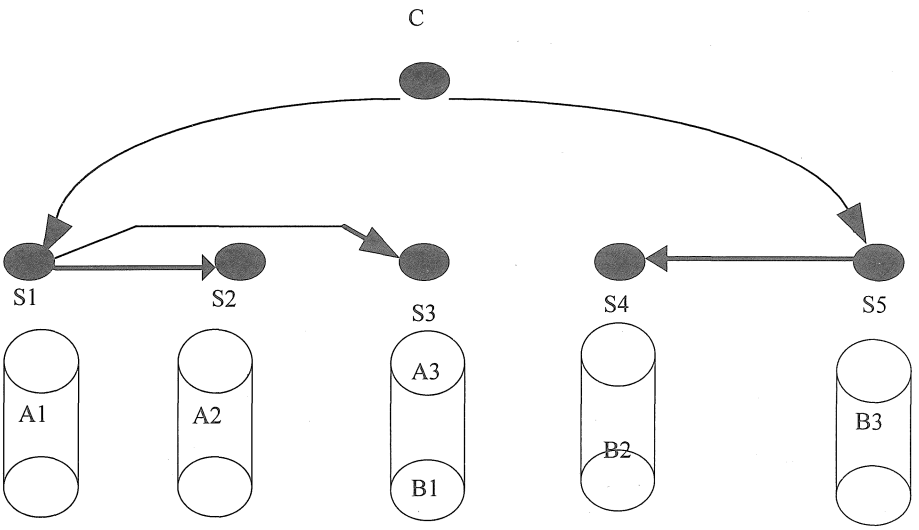


Figura 3: Control de la Replicación Independiente del Control Transaccional

5.3 Seriabilidad y Aislamiento

Un aspecto importante que se debe tener en cuenta al implementar un protocolo de replicación, es el de conservar la seriabilidad de las transacciones que se ejecutan concurrentemente <Coulouris 88>, ya que la existencia de varias copias del mismo dato podría facilitar que varias transacciones accedieran al dato simultáneamente en forma conflictiva.

Si bien esta es una propiedad garantizada por los mecanismos de control transaccional, el diseño de la implantación del protocolo de replicación, debe conservarla.

Para el caso específico en que el control transaccional es independiente del control de replicación, esto se logra mediante el uso de los candados apropiados en cada una de las réplicas. Se garantiza así que para un dato (registro) replicado nunca podrán existir simultáneamente dos coordinadores de operaciones conflictivas pertenecientes a transacciones diferentes.

5.4 Actualización de Réplicas en el Protocolo LAVP

En el protocolo LAVP, no siempre participan todas las réplicas de un dato en las operaciones que lo actualizan, si el quórum de escritura así lo permite. Esto implica que debe contarse con algún mecanismo que ponga al día esas copias que no participaron en un quórum.

En general existen dos aproximaciones para resolver este problema. En una de ellas el proceso encargado de modificar las réplicas, detecta la situación y la registra de manera que la actualización se intente en forma diferida el número de veces que sea necesario o que el sitio de la réplica al recuperarse (si estaba caído) se pueda poner al día haciendo la consulta respectiva.
En la segunda aproximación, la puesta al día se hace en el momento en que una réplica participa en un nuevo quórum, aprovechando el hecho que en dicho quórum siempre debe participar por lo menos una réplica actualizada. El agente que esté controlando el quórum (el coordinador), al detectar que hay réplicas desactualizadas, procede a ponerlas al día.

Para determinar si una réplica está o no actualizada, se maneja un número de versión para cada dato replicado que se va incrementado cada vez que es modificado. El dato más actualizado es el que tiene mayor número de versión.

6. IMPLANTACIÓN DE LOS PROTOCOLOS LAVP Y CPCS EN EL SISTEMA DGDBM

Se escogió el esquema de control de replicación independiente del control transaccional para implantar los protocolos LAVP y CPCS en el sistema DGDBM. A continuación se describe el diseño final implantado.

6.1 El Agente Administrador

Se creó un agente cliente especial con funciones de administración en el sistema DGDBM que entre otras cosas se encarga de la administración de la replicación de archivos, liberando así a los agentes clientes de cualquier función que se relacione con la replicación de archivos.

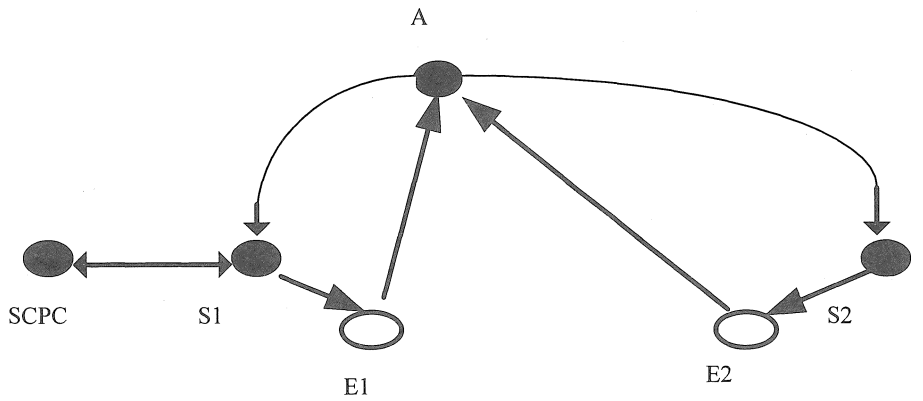


Figura 4: Reserva de Datos y Generación de Esclavos para la Replicación de un Archivo

6.2 Registro del Protocolo LAVP o CPCS

El registro de un protocolo de replicación para un archivo, lo hace el Agente Administrador del sistema, mediante una transacción especial que puede ejecutarse simultáneamente con las transacciones de los usuarios. El proceso consiste de dos etapas: en la primera el Agente Administrador transmite la información relacionada con el protocolo al servidor del sitio actual del archivo. Además se asignan candados sobre la copia primaria del catálogo y sobre el archivo a replicar, y se generan los esclavos que atenderán la transacción. Finalmente el Agente Administrador recibe los mensajes de los esclavos indicándole que en todos los sitios se puede iniciar el registro de la replicación. En la figura 4, se ilustra el intercambio de mensajes que se lleva a cabo en esta etapa, suponiendo que se está replicando un archivo cuyo sitio original es el 1 y que se quiere una réplica adicional en el sitio 2.

En la segunda etapa, el Agente Administrador ordena al servidor del sitio actual del archivo que inicie la replicación del archivo enviando su contenido a los demás sitios, orden que es retransmitida al esclavo respectivo. El esclavo a su vez inicia el envío efectivo de los registros del archivo a los servidores de los sitios en donde haya réplicas. En cada uno de estos sitios se retransmite el registro al esclavo respectivo.

Una vez terminada la retransmisión del contenido del archivo, los servidores involucrados informan el hecho al Agente Administrador, quien continúa con las fases de prevalidación y validación de la transacción.

6.3 Acceso de un Archivo Replicado con LAVP

El uso de un archivo replicado con LAVP dentro de una transacción, tiene tres etapas: iniciación de la operación, consecución de quórum y terminación de la operación.

En la *etapa de iniciación* de la operación el agente cliente ejecuta la lógica propia de cada aplicación y envía la solicitud de ejecución de la operación al servidor coordinador de operaciones para el archivo. Si la operación es **apertura** del archivo, en esta etapa el agente cliente establece el sitio que tendrá el rol de coordinador, es decir, aquel sitio que primero responda a la solicitud de apertura del archivo, siendo éste el único servidor con el que se comunicará para los operaciones con el archivo a manipular.

En la *etapa de consecución de quórum* el servidor coordinador solicita a los sitios en que hay réplicas del archivo, el voto que indica que han colocado el candado apropiado sobre la réplica respectiva. En cada sitio de réplica, la asignación del candado se hace en la misma forma en que se haría la asignación de un candado para un archivo no replicado.

Una vez alcanzado el quórum, se realiza la *etapa de terminación* de la operación en la cual la solicitud de operación se retransmite al esclavo respectivo en el sitio del servidor coordinador, como si se tratara de un archivo no replicado. El esclavo responde directamente al cliente.

Estas tres etapas se repiten para cada operación que se ejecute sobre el archivo. Algunos aspectos importantes para resaltar en este proceso son los siguientes:

- Si no responde un sitio en el que hay un réplica de un archivo que se quiere abrir, el agente cliente intenta con otro en forma recurrente hasta agotar la lista de sitios con réplicas.

- Cada vez que se recibe un voto se registra en una **lista de quórums pendientes** y se verifica si se alcanzó o no el quórum.

- Cuando existe un candado incompatible con el solicitado en un sitio determinado, la transacción se pone en lista de espera en ese sitio. Esto puede implicar que toda la transacción quede en espera si no se alcanza el quórum. El tratamiento de transacciones en espera se hace en forma local como si se tratara de un archivo no replicado.

Cuando termina la transacción dentro de la cual se están ejecutando estas operaciones, el servidor coordinador retransmite a todos los servidores de réplicas que participaron en el quórum las modificaciones registradas.

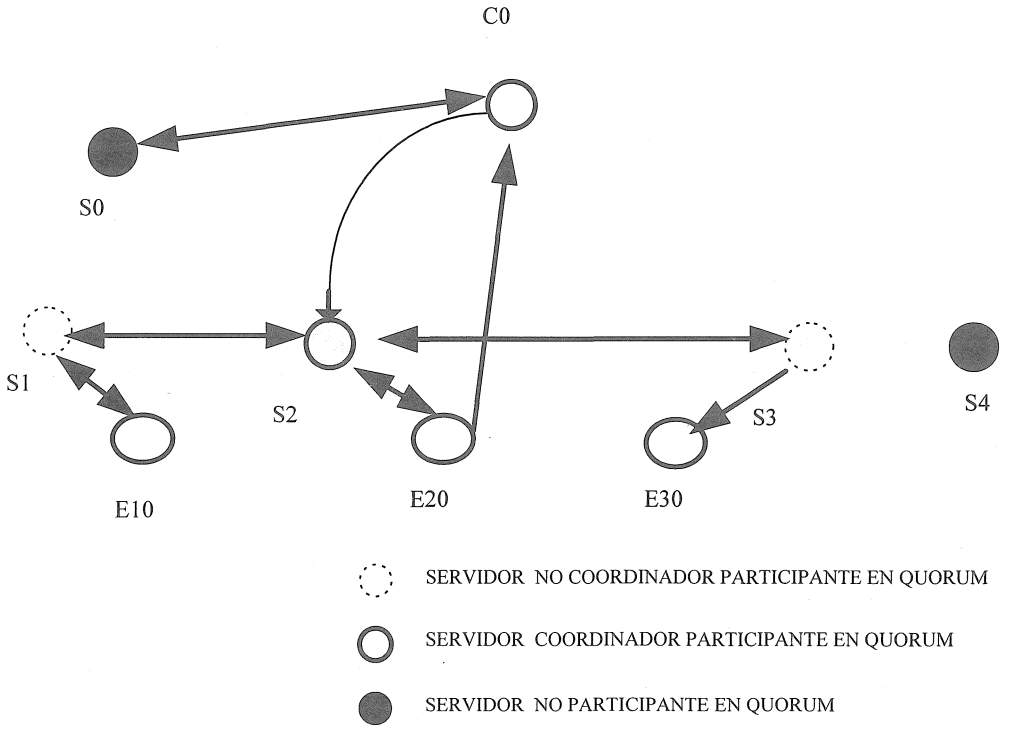


Figura 5: Uso de un Archivo Replicado con Protocolo LAVP

En la figura 5 se ilustra un ejemplo del intercambio de mensajes para el uso de un archivo replicado con LAVP que tiene cuatro réplicas en los sitios 1,2, 3, y 4. Además en esta figura se distinguen los roles de coordinador, participante y no participante en quórum que pueden desempeñar los agentes servidores de los sitios en que hay replicas de un archivo.

6.4 Recuperación de Sitios con Archivos Replicados con LAVP

Desde el punto de vista de la recuperación por caída de un sitio se presentan tres casos: recuperación de un Servidor no participante en quórum, recuperación de un Servidor participante en un quórum pero no coordinador, y recuperación de un servidor coordinador de quórum.
Un sitio no participante en quórum cuyas réplicas se desactualizan, se recupera al participar en nuevos quórums, como se mencionó antes. Un sitio participante en un quórum para modificar un archivo, tiene

una tabla de candados en memoria secundaria que revisa al recuperarse. De esta forma puede detectar si tiene candados sobre réplicas de archivos con protocolo LAVP, caso en el cual debe comunicarse con el servidor coordinador de la operación y resolver la situación validando o anulando la transacción.

Un servidor coordinador de operaciones para un archivo, se comporta ante el control transaccional ejercido por el agente cliente como si tuviera un archivo sin replicar; por lo tanto, si se cae durante una transacción, su recuperación es tratada por la lógica usual de recuperación de transacciones.

6.5 Acceso de un Archivo Replicado con CPCS

En este protocolo las modificaciones se hacen siempre en la copia primaria del archivo. Por lo tanto, para esta clase de operaciones, el archivo se comporta como si se tratara de un archivo sin replicación ubicado en el sitio en el que esté la copia primaria. El agente cliente siempre que manipule un archivo replicado con CPCS, determina el sitio de la copia primaria mediante comunicación con su servidor local y dirige a ese sitio todas las operaciones de modificación.

Una vez terminada la transacción que efectuó las modificaciones, el servidor coordinador las difunde a los sitios en que hay réplicas del archivo. Esta difusión no garantiza, sin embargo, que todas las réplicas sean actualizadas, puesto que algunas de ellas pueden estar en sitios aislados o caídos.

En cuanto a las operaciones de consulta, dado que en este protocolo se pueden hacer en la copia primaria o en las copias secundarias a voluntad del usuario, se ha implementado una función que permite las consultas de registros de la copia secundaria, y otra para hacer las consultas sobre la copia primaria.

6.6 Recuperación de Sitios con Archivos Replicados con CPCS

Un sitio que se cae o se aisla y que tiene archivos replicados con CPCS, puede estar en una de dos situaciones: la réplica que tiene es la copia primaria, o la réplica que tiene es la copia secundaria.

Cuando se presentan fallas en un sitio cuya réplica es la copia primaria del archivo, la recuperación del sitio y del archivo se hace como si se tratara de un archivo sin replicar, es decir, se actúa de acuerdo a la lógica de la recuperación de transacciones. De esta forma se garantiza que la copia primaria del archivo siempre estará actualizada y en un estado correcto y consistente.

Si se trata de copias secundarias cuyos sitios se aislan o se caen mientras se ejecuta una modificación en la copia primaria, éstas se desactualizan y solo se vuelven a ponerse al día cuando, periódicamente y de acuerdo con un criterio preestablecido, el Agente Administrador ejecute la función diseñada con tal fin.

7. CONCLUSIONES

El diseño e implantación de protocolos de replicación de archivos en el sistema DGDBM nos lleva a formular las siguientes conclusiones:

- El manejo de archivos replicados en combinación con archivos no replicados, debe hacerse en forma transparente para el usuario.
- Siempre que un sistema ofrezca protocolos de replicación, también debe ofrecer un esquema de administración de esa replicación.

- La administración de la replicación debe ser independiente de las aplicaciones que manejan los archivos replicados.
- La implementación de un protocolo de replicación debe prevenir la alteración de las propiedades transaccionales del sistema.
- Los aspectos que los protocolos de replicación no contemplan, pero que deben implantarse, requieren de atención especial para que funcionen coordinadamente con el resto del sistema.

En el sistema DGDBM fue posible implantar simultáneamente múltiples protocolos de replicación de archivos, ofreciendo a los usuarios del sistema la flexibilidad de poder decidir el número de copias de cada archivo y el protocolo de replicación asociado. En el futuro, se planea extender estas facilidades llevando estadísticas de uso de los archivos para que el usuario tenga información que le permita decidir dinámicamente el grado de replicación adecuado para un archivo y el protocolo que lo maneja de manera más eficiente.

BIBLIOGRAFÍA

<Abásolo 94> Abásolo, J., Blanco R., Mendieta, F., "Metodología de Diseño de Bases de Datos y su Implantación sobre Oracle 7", Reporte de Investigación Facultad de Ingeniería, Universidad de los Andes, Bogotá-Colombia, 1994

<Ahamad 90> Ahamad, M., Ammar, M., Cheung, S., "Replicated Data Management in Distributed Systems", Readings in Distributed Systems, Casavant and Singhal, editores, IEEE Computer Society Press, 1990.

<Barbara 91> Barbara, L., Gruber, R., Johnson, P., Shrira, L., "A Replicated Unix File System", Operating Systems Review ACM press, vol 25, No. 1, Enero 1991.

<Birman 93> Birman, K., "The Process Group Approach to Reliable Distributed Computing", CACM, vol 36, No. 12, Diciembre 1993.

<Castro 95> O. Castro, "Diseño e implantación de protocolos de replicación de archivos en el sistema ", Tesis de Magister, Departamento de Sistemas y Computación, Universidad de los Andes, Bogotá-Colombia 1995.

<Ceri 84> Ceri, S., Pelagatti, G., "Distributed Databases: Principles and Systems", McGraw Hill Company, 1984.

<Couloris 88> Coulouris, G., Dollimore, J., "Distributed Systems", Addison-Wesley, 1988.

<Franky 95> Franky, M., DGDBM: Programming Support for Distributed Transactions over Replicated Files", ACM Operating Systems Review, Vol 29 No. 3, July 1995, pp 64-74.

<Gray 75> Gray N., Lorie R. A., Putzolu G. R., Traiger I. L., " Granularity of Locks and Degrees of Consistency in a Shared Data Base", IBM Research, 1975.

<Hu 92> Hu, P., Wilbur, S., "Low Storage Cost, Partition-Tolerant Dynamic Algoritms for Replicated File Systems", IEEE, marzo 1992.

<Nelson 93> P.A. Nelson, "GNU dbm (GDBM) versión 1.7", Software libre GNU, disponible mediante frp anónimo al directorio */pub/gnu* del nodo *prep.ai.mit.edu*, Septiembre 1993.

<Rangel 94> E. Rangel, "DGDBM: Ambiente de programación de aplicaciones transaccionales distribuidas sobre archivos de acceso directo", Tesis de Magister, Departamento de Sistemas y Computación, Universidad de los Andes, Bogotá-Colombia 1994.

DS-LED: A DISTRIBUTED APPLICATION FOR THE SIMPLEX METHOD

J. Herrero*, A. Paccanaro*†

*Laboratorio de Electrónica Digital
Universidad Católica "Nuestra Señora de la de Asunción"
Asunción, Paraguay
Tel: +595-21-334650
Fax: +595-21-310587
e-mail: *apaccana@galactica.it*

†Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Milano, Italy

ABSTRACT

A parallel version of the simplex method has been developed and then implemented in a distributed system. This paper presents the distributed application which includes a dynamical load balancing algorithm and is fault tolerant. Finally the results of some tests are presented which show the great advantages in the response time of the parallel algorithm with respect to the linear one.

1. A PRELIMINARY STUDY OF THE PROBLEM

DS-LED implements a variation of the linear programming algorithm (simplex Method) in a distributed environment. The idea of this project was to take advantage from parallel processing in order to optimize the resolution time for this problem.
The reader is supposed to be well acquainted with the main concepts and terminology of the simplex method and distributed systems. For further readings see Prawda[1982], Shivaratri et al.[1992], Bowen et al.[1992], Shieh et al.[1990].

A study of the response time of the linear algorithm for the simplex method was carried out. Experiments were executed on a MicroVAX 3100 machine with Ultrix operating system. Fig.1 illustrates some results. It can be seen that the operation of search of the column vector which has to enter the base - from now on this operation will be called entry - is a very expensive operation when the number of columns is large.

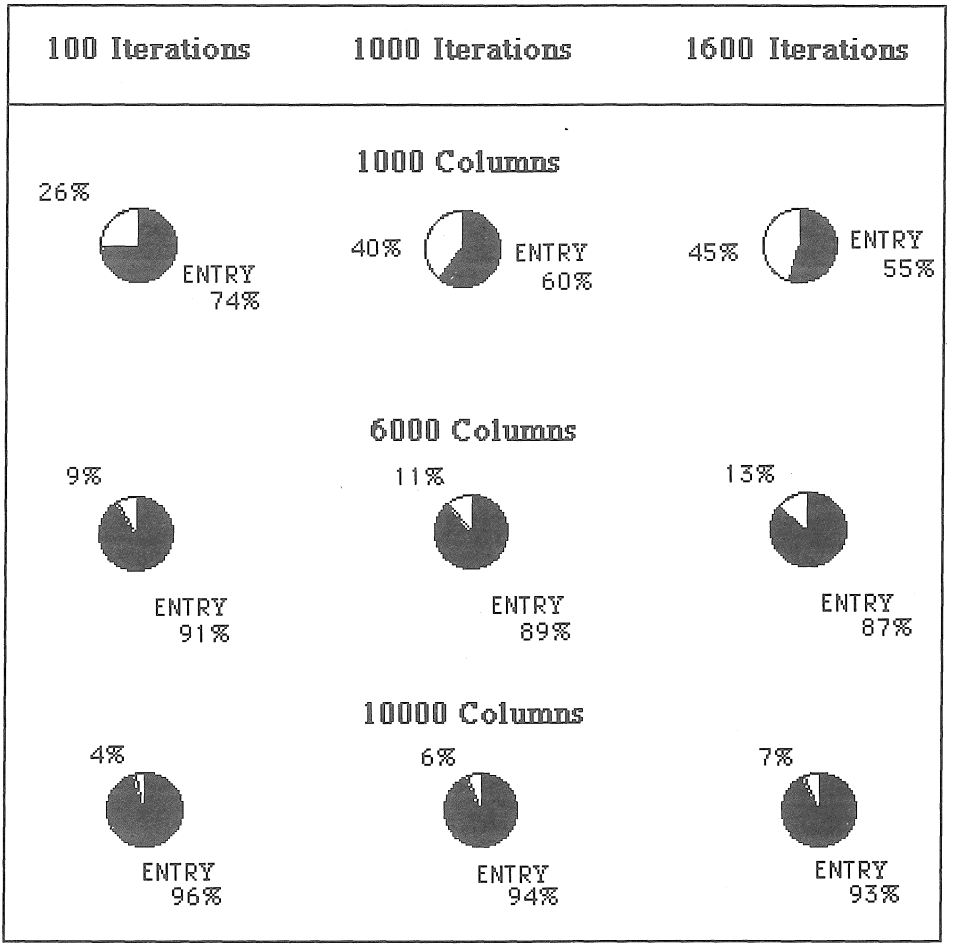


Fig. 1

It was noticed that the entry operation could be parallelized since the operations which have to be carried out in order to find such a vector can be executed in an independent fashion over each column. Therefore it made sense to try which performances could have been obtained with a

parallel version of the algorithm, where different machines could have carried out the entry operation simultaneously and independently.
For this purpose a distributed application called DS-LED has been designed and implemented, and is presented in this paper.

2. SYSTEM IMPLEMENTATION

The project was carried out at the LED -Laboratorio de Electronica Digital - of the Catholic University of Asuncion, Paraguay.
The system was developed using a MicroVAX 3100 with Ultrix operating system with 16 Mb RAM, and 3 PC with SCO Unix operating system, of which 2 were 80486s at 66 MHz and 33 MHz with 8 Mb and 4 Mb RAM respectively and one was a 80386 at 40 MHz with 8Mb RAM; the programming language used was C; communication between processes was realized through sockets interface. The various machine communicated through an Ethernet Net with the TCP-IP communication protocol.

The parallel system implemented is constituted of 2 kind of processes, called client and server respectively.

The number of servers can be greater than one and the various servers cooperate in order to realize the entry operation. To do this the matrix of the elements which are not in base will be partitioned in various disjunct sets, and each one of them will be given to a distinct server.

On the other hand there will exist only one client which will have to carry out the non-parallelizable part of the algorithm.
Therefore the client will have to keep a complete copy of all data.

All the processes will be run on independent machines in order to take full advantage of the parallel processing of the data.
Main features of the application are a load balancing and fault tolerance algorithm, which are presented in the following sections.

a) Load Distribution

In order to obtain better response time the amount of columns which has to be distributed to each server must take into consideration both the speed of the machine on which the server is running and the number of processes which are already present on it.

Moreover since the amount of processes being executed by the various servers may vary during time, a dynamical load balancing should be carried out in order to obtain better performances.
DS-LED implements such load balancing in the following way.
At the beginning of the execution the client waits from each server for a message containing information about the speed of the machine and the number of processes executing on it. Then it can distribute the columns among the various machines according to these measures. It is necessary to point out that the information relative to the speed of each processor is a static coefficient, empirically assigned "a priori" to each processor. Nevertheless it has proven to be extremely important in order to obtain a better response time from the parallel algorithm. Fig. 2 shows the difference in response time obtained introducing such coefficients.

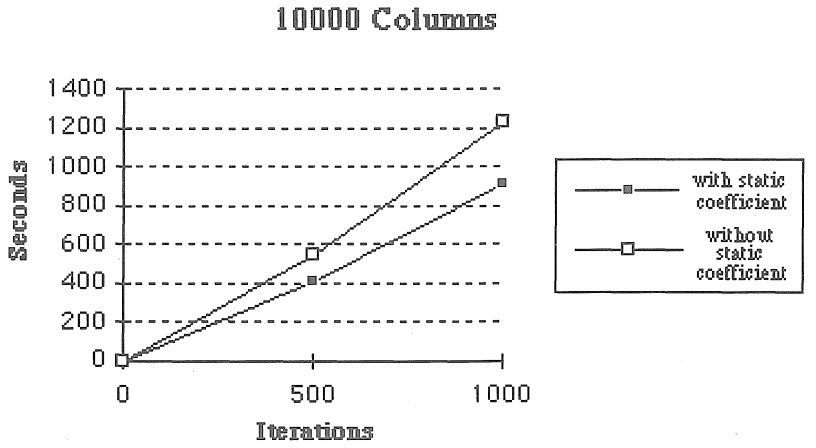


Fig. 2

During execution the client at each iteration of the algorithm measures the response time of each server and accordingly to it, periodically redistributes some columns among the various processor, if necessary.
Fig. 3 shows the performances of the parallel algorithm with dynamical load balancing as compared to the same algorithm without it. The size of the problem was of 10000 columns and there were no variations on the load of each server. The dynamical load balancing algorithm transferred columns among servers between the 700th and 800th iteration.

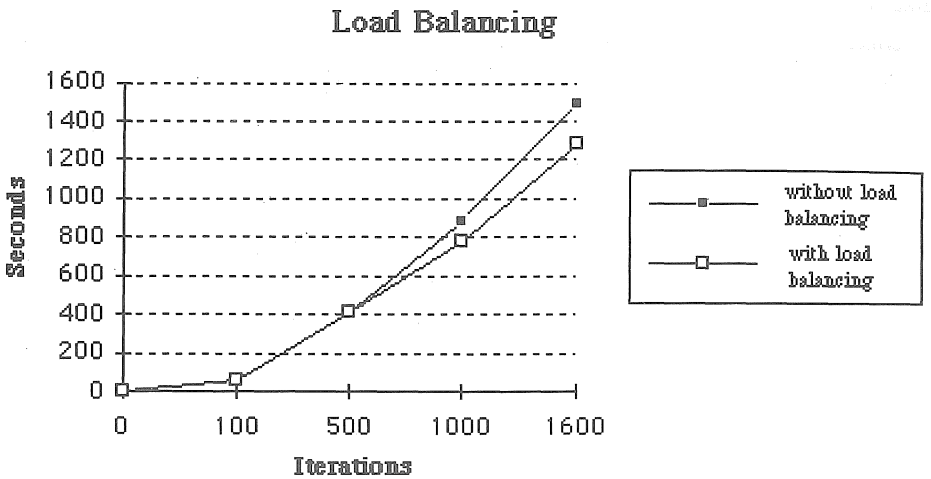


Fig. 3

b) Fault Tolerance

The fault tolerance system implemented in DS-LED deals with the case in which a machine is completely lost. This implies the loss of all data and processes which are being executed on it. Moreover the fault of a machine can cause problems to the whole system if other machines are waiting for some message from it. In order for the system to recover from such a fault it is necessary to (Knight [1987]):

i) detect the fault
ii) detect the damages due to the fault
iii) select and execute an appropriate answer

The actual version of DS-LED can only deal with a complete loss of one or more servers; it can not recover the loss of the client.
Let us see how each of the step mentioned above are implemented.

i) fault detection

In order to detect that a fault occurred to a machine belonging to the system 2 processes were implemented: the Client Fault Tolerant process (CFT) and the Server Fault Tolerance process (SFT).
The SFT process periodically sends a message to the CFT process, which immediately sends back an acknowledgement message. This message is used by the SFT in order to be sure that the

machine where the client is being executed is still alive. On the other hand the CFT process keeps a table containing the addresses of the machines where the servers are executed which is updated each time that a message is received from a SFT process. Periodically the CFT process checks this table in order to see if a server process died. Once a fault has been detected the CFT and SFT processes communicate the problem to the client and server processes respectively.

ii) damage detection

In order to determine the damages, the client process keeps a table of how the columns were distributed over the various servers. In this way he can know which columns were lost due to the fault of one or more servers. With this information it is possible to decide which action to take in order to recover the fault.

On the other hand the server does not need to keep a similar table since, as mentioned before, this version of DS-LED can not recover the loss of a client.

iii) selection and execution of an appropriate answer

Case 1 - fault of the client

If the client dies, the server process will be waiting indefinitely for a message from it. In order to avoid this problem when the SFT detects the fault of the client it sends to the server process a message, it can select which action to take. In this version of the system the only possible answer will be to abort.

Case 2 - fault of the server

If a fault occurs to one or more servers, the client has 2 options: to abort or to redistribute the columns which had been given to the dead server processes.

The abort option is executed when there are no more server processes available. If some server is still alive the client process will redistribute all the columns belonging to the dead processes to the servers which remain alive.

3. SYSTEM EVALUATION

The system was tested in order to measure the advantage in response time obtained with the parallel algorithm with respect to the linear one.

The performance of the parallel algorithm was analyzed while varying the number of columns, since the performance of the linear algorithm depended on the number of columns in the problem.

In the following (fig. 4, 5 and 6) some diagrams are shown representing the time responses obtained for 1000, 4000 and 10000 columns. Response time were taken after 100, 1000, and 1600 iterations.

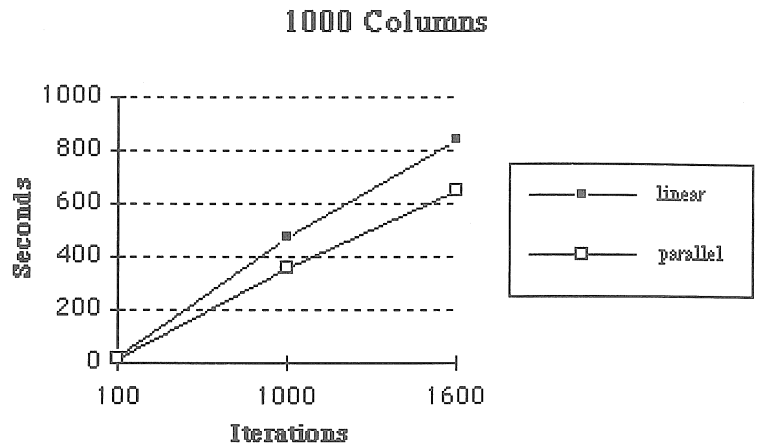


Fig. 4

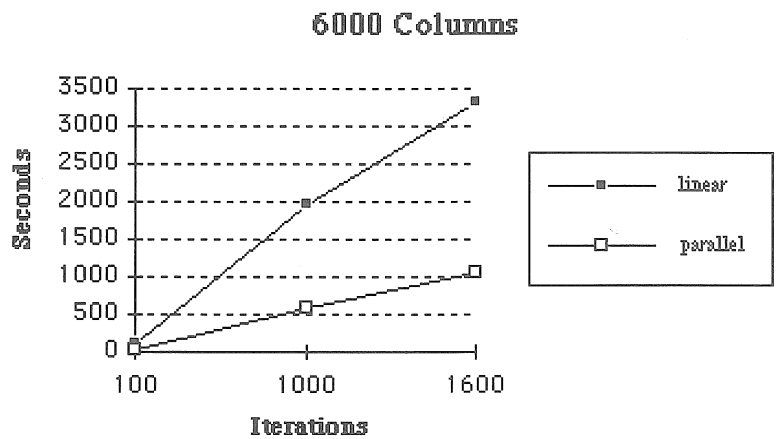


Fig. 5

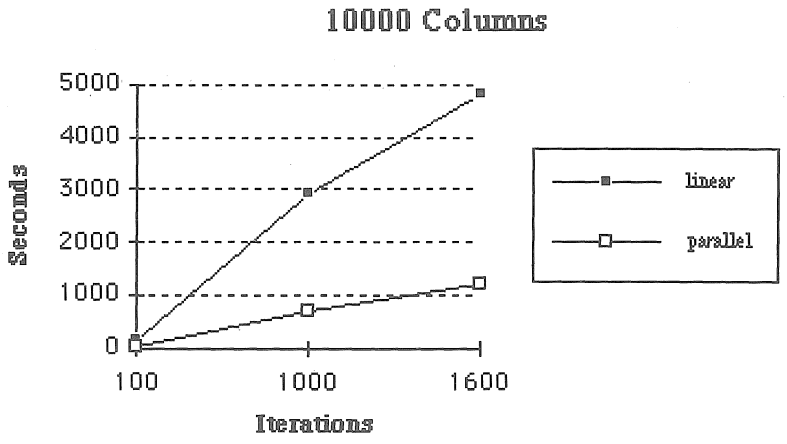


Fig. 6

The advantage obtained using a parallel system are greater for larger sizes of the problem. This can also be seen analyzing the numerical values of the tests presented above which are shown in the following table:

	500 Columns	1000 Columns	2000 Columns	4000 Columns	6000 Columns	8000 Columns	10000 Columns
Linear	635	839	1193	2398	3323	3822	4818
Parallel	637	650	616	901	1056	1158	1203

Table 1

An interesting measure of the advantage in response time obtained with the parallel algorithm with respect to the linear one is given by an the acceleration index. This index is calculated through the following formula:

$$\text{acceleration} = \frac{T_l}{T_p}$$

where t_l and t_p are the time consumed by the linear and by the parallel algorithm respectively
Fig. 7 shows a graph of the acceleration obtained for the tests presented above. The numerical values were taken after 1600 iterations, while varying the size of the problem.
It is important to notice that the acceleration is greater when the size of the problem increases and it reaches the value of 4 for 10000 columns.

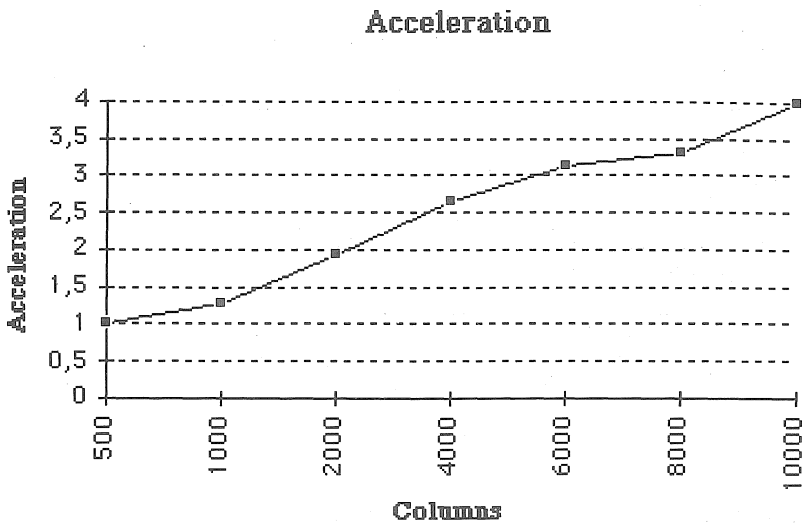


Fig. 7

The increase in the acceleration which is obtained when increasing the size of the problem is due to the fact that the serial section of the program remains constant while the parallelizable section increases when the number of columns increases.

Finally, Fig. 8 presents a comparison of the time used by the 2 algorithms to iterate 1600 times on problems of different size. It is important to point out that the increase in the execution time is much slower for the parallel algorithm.

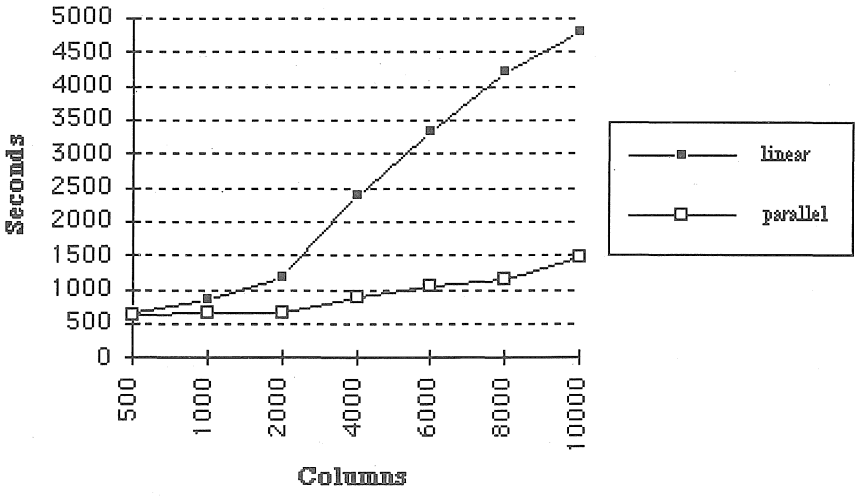


Fig. 8

4. CONCLUSIONS AND FURTHER DEVELOPMENTS

We presented the implementation on a distributed system of a parallel version of the simplex method. The application includes a dynamical load balancing algorithm and is fault tolerant with respect to the loss of one or more servers. The results presented show a great advantage in the response time of the parallel algorithm with respect to the linear one. We are now thinking of including a fault tolerant algorithm which would be able to deal with the loss of the client.

REFERENCES

Niranjan G. Shivaratri, Phillip Krueger and Mukesh Singhal [1992], "Load Distributing for Locally Distributed Systems", COMPUTER, pp. 33-44, DECEMBER 1992.

N.S. Bowen, C.N.Nikolau y A.Ghaffour [1992], "On the Assigment Problem of Arbitrary Process Systems to Heterogeneus Disributed computer Systems", IEEE TRANS. ON COMPUTERS, VOL. 41, pp. 257-273, MARCH 1992.

J.C. Knight and J.I.A. Urguhart [1987], "On the Implementation and Use of ADA on foult-tolerant Distributed Systems", IEEE TRANS. ON SOFT. ENG, VOL SE-13, pp 553-563, MAY 1987.

Y.B. Shieh, D. Ghosal, P.R. Chintamaneni, and S.K. Tripathi [1990], "Modeling of Hierarchical Distributed Systems with Fault-Tolerance", IEEE TRANS. ON SOFT. ENG, VOL 16, pp. 444-457, APRIL 1990.

Juan Prawda [1982], "Métodos y Modelos de Investigación de Operaciones", Limusa, 1982.

Una Experiencia de Programación Distribuida utilizando Paradise

Marcelo Mejía Olvera
Instituto Tecnológico Autónomo de México
Río Hondo No. 1, Tizapán San Ángel, 01000 México, D.F.
e-mail: marcelo@lamport.rhon.itam.mx

Resumen

Paradise es un lenguaje paralelo de alto nivel que permite desarrollar fácil y rápidamente aplicaciones distribuidas. La evaluación del desempeño de un sistema real que implementa el algoritmo de visualización *Ray Tracing* en paralelo muestra que es posible obtener factores de aceleración altos y una eficiencia cercana al 100% utilizando el ambiente de programación Paradise.

Palabras clave: Lenguajes de programación distribuida, Algoritmos de visualización.

1 Introducción

Las gráficas por computadora son una de las áreas de la Computación que requieren realizar mayor cantidad de operaciones matemáticas. Una aplicación real de cálculo intensivo en esta área surge en el proyecto "Reconstrucción Tridimensional de Núcleos Celulares de Cromatina" de la Facultad de Ciencias de la UNAM, en el que se reconstruyen computacionalmente, y se despliegan para llevar a cabo un análisis visual certero, imágenes en tres dimensiones provenientes de la digitalización de fotografías de cortes ultrafinos de núcleos de cromatina. La parte correspondiente a la reconstrucción de la imagen sobre un monitor de dos dimensiones representa un grave problema ya que los algoritmos de visualización eficaces necesitan mucho tiempo para ejecutarse.

En este artículo se describen los resultados obtenidos al utilizar una arquitectura paralela que permite incrementar la velocidad de ejecución de un algoritmo de reconstrucción mediante la distribución de la carga de procesamiento entre un conjunto de estaciones de trabajo conectadas en red. En la sección 2 del artículo se introduce brevemente el algoritmo de reconstrucción utilizado y en la sección 3 se describe la implementación paralela realizada del algoritmo utilizando el lenguaje Paradise. En la sección 4 se presentan las medidas de desempeño global del sistema de visualización, mientras que en la sección 5 se presentan las conclusiones del trabajo.

2. Algoritmo de reconstrucción

Para reconstruir una imagen sobre una pantalla de computadora a partir de los cortes digitalizados se optó por implementar en paralelo el algoritmo *Ray Tracing*, que es uno de los más eficientes para

desplegar imágenes tridimensionales en dispositivos bidimensionales de manera realista. El funcionamiento del algoritmo de *Ray Tracing* está basado en principios de óptica geométrica y considera la interacción de rayos de luz con la imagen tridimensional o escena. El algoritmo básico lanza rayos de luz desde el punto de vista que se desea reconstruir hacia adentro de la escena a través de cada uno de los pixeles de una pantalla virtual; estos rayos viajan en dirección contraria a la propagación de los rayos de luz en la realidad (Green, 1991). Para cada rayo lanzado se determina cuál de todas las superficies de la escena es la más cercana a la posición del observador y por lo tanto es la superficie visible en el pixel correspondiente. Para obtener imágenes realistas que incluyan efectos de sombra, reflexiones y refracciones causados en la realidad por rebotes de rayos de luz, el algoritmo normalmente se extiende lanzado rayos secundarios a partir de cada intersección de un rayo primario y una superficie. Hacia cada fuente de luz de la escena se dirigen rayos de sombra para determinar la contribución de éstas a la iluminación del punto de intersección. Dependiendo de las características de las superficies también pueden lanzarse rayos de reflexión o refracción que, si intersectan con otras superficies, pueden también contribuir a la iluminación del punto de origen de estos rayos. Para calcular la iluminación final de un pixel se consideran las intensidades de todos los rayos creados de manera recursiva que influyen en el punto correspondiente de la superficie visible.

Dado que el algoritmo de *Ray Tracing* requiere de un tiempo de procesamiento muy grande, para aumentar su velocidad de ejecución pueden aplicarse restricciones al algoritmo y/o utilizarse técnicas de aceleración. De acuerdo al grado de realismo necesario en la imagen resultante, se decidió que en nuestra implementación del algoritmo podían aplicarse las siguientes restricciones:

- la escena contiene únicamente dos fuentes internas de luz,
- sólo se toman en cuenta dos rebotes para cada rayo primario, y
- se descartan las texturas.

La idea general detrás de las técnicas de aceleración es reducir el número o el costo de las pruebas de intersección de los rayos con las superficies en la escena. En la implementación realizada se utilizó la técnica de voxelización para reducir el número de pruebas de intersección. En esta técnica primeramente se representa la escena como un conjunto de pequeños cubos (voxeles). A continuación, para cada rayo se determina el voxel que contiene la superficie visible y se realizan las pruebas de intersección (sólo) sobre los objetos contenidos en el voxel. Los cálculos de las pruebas de intersección se simplifican en nuestra implementación ya que se considera que los objetos de la escena están formados por esferas pequeñas, que se adaptan a los bordes redondeados de las células. Para disminuir aún más el número de pruebas de intersección la imagen se preprocesa antes de iniciar el algoritmo de *Ray Tracing* para determinar qué elementos constitutivos de cada voxel contienen una esfera y cuáles están vacíos.

En resumen podemos decir que el algoritmo de *Ray Tracing* implantado recibe la información de la escena tridimensional como entrada, la procesa tomando en cuenta las restricciones y las técnicas de aceleración antes mencionadas, y como salida genera un archivo que indica la iluminación correspondiente a cada pixel del monitor.

3. Programación paralela utilizando Paradise

Para implementar el sistema paralelo de visualización se escogió el lenguaje Paradise (SCA, 1994) que representa la tercera generación de un conjunto de lenguajes de programación paralela derivados de Linda (Carriero and Gelernter, 1990). El funcionamiento de estos lenguajes está basado en el uso de una memoria virtual compartida, denominada espacio de tuplas (ET), que es un espacio de datos disponible a cada proceso de una aplicación distribuida. En el ET se almacenan conjuntos ordenados de valores denominados tuplas. La identificación de las tuplas no se realiza mediante direcciones sino mediante un nombre lógico constituido por un subconjunto de sus valores.

El modelo Linda añade al lenguaje C las operaciones de coordinación entre procesos `out()`, `read()` e `in()` para acceder al ET y permitir la comunicación y sincronización entre procesos. La operación `out(t)` permite agregar la tupla `t` al ET; el proceso que invoca la operación continúa su ejecución inmediatamente. La operación `read(s)` lee una tupla `t` del ET que concuerde con `s` (su nombre lógico sea el mismo) y asigna los parámetros actuales de `t` a los formales de `s`. Si no existe en el ET una tupla `t` que concuerde con `s` entonces el proceso que invoca la operación se suspende hasta que se encuentre una concordancia. La operación `in(s)` es similar a la operación `read(s)`, pero retira del ET la tupla `t` que concuerde con `s`. Existen también, en el modelo Linda, las operaciones `inp(s)` y `readp(s)` que son versiones no bloqueantes de `in(s)` y `read(s)`. En Paradise se cuenta, además, con operaciones para crear, abrir, cerrar y destruir espacios de tuplas persistentes.

Paradise es particularmente útil en el contexto de programas distribuidos organizados como una colección de trabajadores idénticos. En este modelo un programa no se particiona en tareas sino que se copia *n* veces, donde *n* está determinado por el número de procesadores disponibles. Un proceso capataz introduce en ET los trabajos a realizar y cada uno de los trabajadores busca continuamente en ET un trabajo para ejecutar. El modelo de trabajadores replicados presenta las ventajas de ser fácilmente escalable en función del número de procesadores disponibles y de no necesitar ninguna interacción directa entre los trabajadores.

En el sistema de visualización paralelo desarrollado se divide la pantalla de la computadora donde se despliegan los resultados en 1600 segmentos rectangulares de 12x12 pixeles cada uno. Se utilizan dos ET en la aplicación. El primero contiene en cada instante el número del siguiente segmento que debe ser procesado. Cada trabajador accede repetidamente al ET_1 para leer (`in`) y actualizar (`out`) este número, y procesa el segmento correspondiente utilizando el algoritmo de *Ray Tracing* para calcular la iluminación de cada uno de sus 144 pixeles (Mejía et al, 1994). Los trabajadores almacenan los resultados del procesamiento de cada segmento en el segundo ET.

El sistema de visualización se ejecuta en un conjunto de estaciones de trabajo RS6000 de IBM y consta de tres etapas principales:

- Preprocesamiento de la imagen
- *Ray Tracing* paralelo
- Desplegado

El funcionamiento del sistema es controlado desde una computadora personal que se encarga inicialmente de preprocesar la imagen, copiar (utilizando rcp) la imagen preprocesada a las estaciones de trabajo del sistema y lanzar (mediante rsh) la ejecución en paralelo de los procesos Paradise que ejecutan el algoritmo paralelo de *Ray Tracing*. Posteriormente, la computadora personal colecta los resultados del algoritmo de reconstrucción que indican la iluminación que debe tener cada uno de los pixeles de la pantalla y despliega la imagen sobre su monitor.

Para maximizar el paralelismo real del sistema, en cada computadora que participa en el algoritmo de *Ray Tracing* paralelo se ejecuta solamente un trabajador. El número de segmentos que procesa cada trabajador es variable, ya que depende de la cantidad de cálculos que necesite el algoritmo de *Ray Tracing* secuencial para determinar la iluminación de los pixeles que le son asignados.

4. Evaluación del desempeño del sistema

Las pruebas realizadas para evaluar el desempeño del sistema paralelo se efectuaron utilizando una imagen típica que nos proporcionó la Facultad de Ciencias de la UNAM y se efectuaron principalmente para contestar dos preguntas: 1) ¿es posible reducir a menos de una hora el tiempo necesario para reconstruir una imagen típica completa (con el realismo deseado) utilizando el sistema paralelo?, y 2) ¿la eficiencia del sistema desarrollado se mantiene alta conforme se aumenta el número de procesadores? La respuesta a la segunda pregunta es crucial ya que si la eficiencia del sistema no se degrada al incrementar el número de trabajadores, entonces puede reducirse el tiempo de procesamiento simplemente aumentando el número de estaciones de trabajo que conforman el sistema.

Antes de presentar los resultados experimentales obtenidos al medir el desempeño del sistema paralelo, definamos las siguientes variables:

- n número de trabajadores que ejecutan el algoritmo,
- Tn tiempo de ejecución del algoritmo usando n trabajadores,
- $s = T1/Tn$ aceleración (relativa) del sistema
- $e = Tn/n$ eficiencia del sistema paralelo

En un sistema paralelo se espera que Tn disminuya al aumentar n y se busca que la aceleración s obtenida sea una función lineal de n, y por lo tanto que la eficiencia e esté cerca del 100%. Las figuras 1 y 2 muestran la aceleración relativa y la eficiencia del sistema paralelo en función de n para subconjuntos de la pantalla de 16 y 160 segmentos, respectivamente. El valor de n está limitado a 9 por el número de licencias disponibles de Paradise

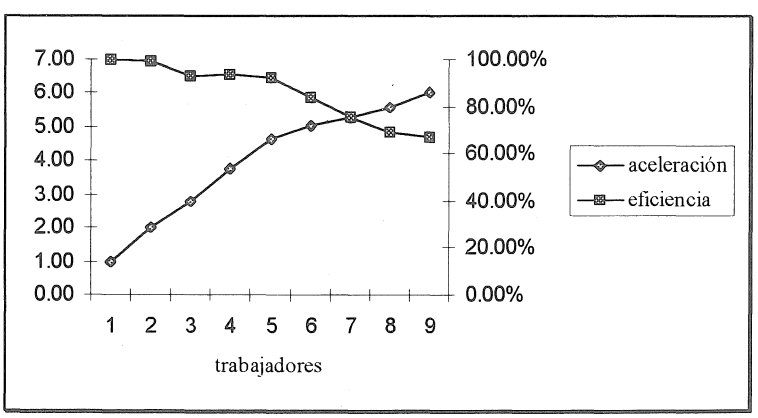


Figura 1. Aceleración y eficiencia para 16 segmentos.

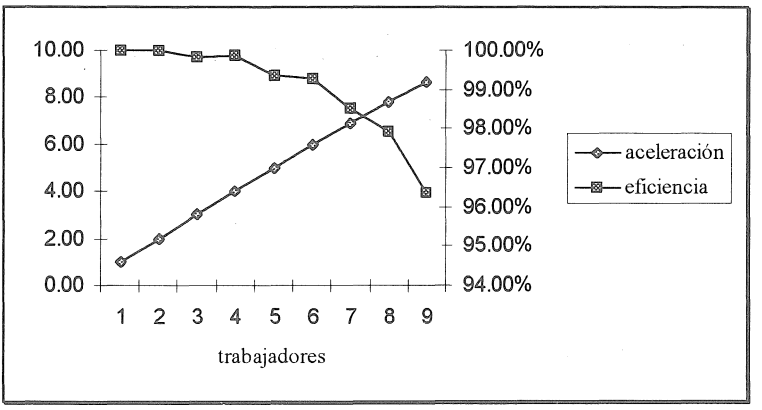


Figura 2. Aceleración y eficiencia para 160 segmentos.

Debe notarse que *T1* es un poco mayor que el tiempo necesario para correr el algoritmo de *Ray Tracing* de manera secuencial (*Ts*) ya que el trabajador consume tiempo al acceder tres veces a los espacios de tuplas (para obtener y actualizar el número del siguiente segmento a procesar, y para escribir el resultado del procesamiento) por cada segmento que debe procesar. Por lo tanto, la aceleración absoluta (*Ts/Tn*) es menor que la aceleración relativa mostrada en las figuras 1 y 2. Por ejemplo, para 16 segmentos y dos trabajadores la aceleración disminuye de 1.99 a 1.76 (*Ts* = 265s, *T1* = 299s y *T2* = 150s). De cualquier forma, la implementación paralela con dos trabajadores es más rápida que el algoritmo secuencial, y la velocidad absoluta aumenta al agregar más trabajadores.

En las figuras 1 y 2 puede observarse que al aumentar el número de segmentos de 16 a 160, la eficiencia del sistema se degrada con mayor lentitud. Esto se debe a que al aumentar el tamaño del

problema, disminuye la proporción del tiempo total del algoritmo paralelo durante el cual existen computadoras sin trabajo al agotarse los segmentos a procesar. Podemos afirmar entonces que si se escogen adecuadamente la complejidad del algoritmo de *Ray Tracing*, la granularidad de la segmentación y el número de trabajadores, el sistema de visualización paralelo es capaz de obtener valores importantes de aceleración y eficiencia.

Utilizando el máximo de 9 trabajadores, el sistema paralelo procesó los 1600 segmentos correspondientes a una pantalla completa en 2894s (menos de 49 minutos). Dado que $T1$ = 2987s para 160 segmentos y $T9$ = 2894s para 1600 segmentos, podemos afirmar que el desempeño de la implementación paralela se escala bien con el tamaño del problema y el número de trabajadores dentro de los límites de nuestras pruebas.

6. Conclusiones

El uso del paralelismo en computación es de gran utilidad en aplicaciones que requieren grandes cantidades de cálculos matemáticos. En el campo de la graficación tradicionalmente se han utilizado supercomputadoras vectoriales, como la CRAY YMP, o máquinas masivamente paralelas, como la CM-5. El trabajo presentado en este artículo muestra que es posible utilizar estaciones de trabajo sencillas conectadas en red para implantar algoritmos, como el *Ray Tracing*, que consumen mucho tiempo de procesador. En este sentido se observa la eficacia de la programación paralela en una ambiente distribuido.

El algoritmo de *Ray Tracing* puede paralelizarse de distintas maneras. En la implantación realizada se consideró un paralelismo de grano burdo que se adapta bien al ambiente de ejecución distribuido. La repartición de la carga de trabajo en términos de grupos de pixeles que constituyen segmentos de la imagen, y la posibilidad de variar el tamaño de los segmentos en función de la complejidad de la escena, permiten obtener una buena relación entre el tiempo que cada computadora dedica al procesamiento de la imagen y el tiempo que emplea para comunicarse (indirectamente a través del ET) con otros procesos.

El uso de un lenguaje de programación paralelo de alto nivel como Paradise facilita el trabajo de desarrollo de aplicaciones distribuidas y disminuye el tiempo de diseño, implementación y pruebas comparado con el uso de llamadas de bajo nivel al sistema como son los sockets.

En Paradise no existe la operación `eval("nombre",f())` de Linda que permite crear procesos dinámicamente. En el sistema de visualización paralelo los procesos son creados por medio de un proceso externo a Paradise. Para automatizar la creación de procesos y realizarla al interior de un proceso Paradise, se han creado en el ITAM recientemente librerías que hacen posible ejecutar procesos Paradise en máquinas específicas; estas librerías se encargan de copiar el código ejecutable en la máquina remota y de lanzar la ejecución del proceso.

Referencias

Carriero, N., and Gelernter, D. (1990); How to Write Parallel Programs - A First Course; The MIT Press.

Green, S. (1991); Parallel Processing for Computer Graphics; The MIT Press.

Mejía, M., Hernández, H., y Quiroz, J. (1994); Visualización Tridimensional en Paralelo; Congreso Nacional de Informática, Fundación Arturo Rosenblueth, México, D.F.

SCA (1994); Paradise - User's Guide and Reference Manual; Scientific Computing Associates.

PRIORITY INVERSIONS IN RATE MONOTONIC 8802/4: ANALYSIS OF THE WORST CASE

R. Cayssials, J. Orozco, J. Santos and E. Ferro
Dep. Ingeniería Eléctrica, Instituto de Ciencias e Ingeniería de Computación
Universidad Nacional del Sur, 8000 Bahía Blanca, Bs. As., Argentina
E-mail: {iecayss, ieorozco, iesantos, ieferro}@criba.edu.ar
FAX: 00 54 91 553122

I. Introduction

The ISO 8802/4 (Token-Bus) Standard offers an optional priority mechanism intended to operate in a real-time environment, in which messages generated at one node must be transmitted before a certain deadline. Although 8802/4 is able to implement a Fair Round-Robin priority discipline [2, 7], it can not implement a strict Rate Monotonic (RM) discipline because, in certain circumstances, *priority inversions* take place. In those cases, the use of the transmission medium is granted not to the node with the highest priority but to some lower priority node. In order to determine the LAN's schedulability it is necessary to know the worst case of priority inversions that each node may suffer. The purpose of this paper is to present a formally proved set of lemmas and theorems that, applied to a Real-Time Token-Bus LAN operating under the *Rate Monotonic* priority discipline, allows the determination of the number of inversions in the worst case and, therefore, its schedulability.

II. The Empty-Slots Method

Several methods to determine the schedulability of a real-time system have been proposed [3, 4, 5, 8]. In this paper the *empty-slots method* [8] will be used.

S(m) denotes a network of m nodes, named 1, 2, ...,m. In the empty-slots method, time is assumed to be slotted. During the duration of one slot, called *slot-time* and symbolized T_s, the transmission medium is allocated to one and only one node with no preemptions allowed during the slot-time.

Slots are notated t and numbered 1, 2, ... For the schedulability calculations, the slot-time is taken as unit of time. When all pending messages have been transmitted and no new messages are generated, empty-slots appear. $e_{j(m)}$ denotes de j^{th} empty-slot in the execution of S(m).

In [6] it has been proved that a system S(m) in which nodes may suffer up to I inversions is RM schedulable iff

$$\forall i \in \{2, ..., m-I\} \qquad T_i \geq e_{(I+1)(i-1)}$$

where $e_{(I+1)(i-1)}$ denotes the $(I+1)^{th}$ empty slot in S(i-1). It is proved to be

$$e_{(I+1)(i-1)} = \text{least } t | t = 1 + I + \sum_{h=1}^{i-1} \left\lceil \frac{t}{T_h} \right\rceil$$

where $\lceil\ \rceil$ symbolizes the monadic ceiling operator. In the scheduling analysis that follows, t_0 (i) denotes the instant at which the token abandons node_i and a simultaneous generation of messages takes place at node_i and all downstream nodes following a round without transmissions.

III. The Token-Bus Priority Mechanism

The right of access to the physical medium is given by the possession of a token which is passed by stations residing in the medium. As the token is passed from a predecessor station to its successor, a virtual ring is formed on the real bus.

The 8802/4 priority mechanism belongs to the type that can not implement a strict RMS [9]: It is obvious, for instance, that if a simultaneous generation of messages of different priority takes place after a round with no transmissions, the use of the medium will be randomly granted to the first node receiving the token irrespective of its priority.

The Standard [1] provides an optional mechanism with four levels of priority, called *access classes*. In each station there are four request queues corresponding to the four priority levels; the bandwidth is allocated by timing the rotation of the token around the ring. Messages of the higher priority (class_6) at any station are transmitted always with the only proviso that a certain time, called *High Priority Token Holding Time*, *HPTHT*, is not exceeded.

Each of the three lower classes is assigned a *"Target" Token Rotation Time*, symbolized *TTRT*. For each class the station measures the time it takes the token to circulate around the logical ring. If it returns in less than the target time, the station can transmit messages of that class until the time has expired. Everything happens as if high priority stations had an infinite *TTRT*.

When the token arrives in the node, its *TTRT* is loaded in a counter and a regressive count is started. The node will be able to transmit if at the next token arrival the counter content is not zero. Since the unit of time is the slot-time and one message is transmitted in each slot, to talk about units of time or about messages heard by the node since the last token reception amounts to the same thing. $R(i)$ and $\widetilde{R}(i)$ shall denote the *TTRT* of node_*i* and the $\max_{i<k\leq m}\{R(k)|\, R(k)<R(i)\}$, respectively.

Messages of the same priority within one real (physical) station may be considered to be generated at one *virtual node*. Therefore, a real station can have up to four virtual nodes. In what follows, the number of virtual nodes in the system will be taken as its number of nodes.

IV. Rate Monotonic Scheduling

RM scheduling is a *de facto* standard, supported, among others, by the USA Department of Defense. In it, conflicts of access to the medium are solved by prioritizing the nodes according to monotonically increasing periods. Some additional rule must be provided in order to break ties when nodes have the same period.

The proof of necessity of the ordering rules as well as the formal proof of all the lemmas and theorems that follow are omitted in this summary but will be included in the full paper.

In what follows let us consider a Token-Bus LAN in steady state assembled under the general assumptions described at the Introduction for the empty-slots method, plus the following particular ones:

a) If a station contains a virtual node belonging to class_6, no other virtual nodes are allowed in it.

b) Stations containing nodes belonging to lower classes may have up to three virtual nodes, provided the following ordering rules are fulfilled.

A. Ordering Rules

Rule 1: Nodes are placed in the logical ring ordered, in the token sense of rotation (e.g. clockwise), by monotonically decreasing *TTRT*s.

Rule 2: If $T_i < T_j$, $R(i) \geq R(j)$.

Rule 3: From Rules 1 and 2, it follows that if node_*i* precedes node_*j* and $T_i=T_j$, then $R(i) \geq R(j)$.

Monotonically increasing periods are therefore translated into monotonically decreasing TTRTs. Since the 802.4 priority mechanism is supposed to associate higher priorities to larger TTRTs, the mechanism would implement RMS. The tie between nodes with the same TTRT is

broken by their placement in the ring: a node preceding another one in the ring is supposed also to have precedence in the right to transmit.

For a given node_*i*, downstream nodes are *i*+1 to m, and upstream nodes, 1 to *i*-1. When in spite of node_*i* having a queued message, downstream nodes transmit before it, inversions take place. For nodes belonging to the lower classes, inversions may come from nodes having the same TTRT or from nodes having smaller TTRTs. They will be called placement and token inversions, respectively. Nodes belonging to the upper class do not have associated TTRTs but, all the same, they may suffer inversions from nodes of the same or lower classes.

B. Scheduling analysis for high priority (class_6) nodes

The number of nodes in class_6 shall be denoted n_6. The period of a class_6 node must be such that, after $t_0(i)$, it can tolerate the worst case of inversions plus the transmission of messages by the upstream class_6 nodes. Inversions come from downstream class_6 nodes and from nodes belonging to lower classes receiving the token before their *TTRT*s have expired.

Therefore

$$\forall i = 1, 2, \ldots, n_6 \qquad T_i \geq n_6 + \sum_{h=n_6+1}^{m} f(h,i)$$

where

$$f(h,i) = \begin{cases} 1, & \text{if } R(h) > n_6 - i + \sum_{x=n_6+1}^{h-1} f(x,i) \\ 0, & \text{otherwise} \end{cases} \qquad (1)$$

C. *Scheduling analysis for lower classes nodes*

Since the priority of a lower class node depends only on its placement and on its *TTRT*, it is irrelevant to which of the lower classes it belongs. In every case, its period must be such that, after $t_0(i)$, it can tolerate transmissions by upstream nodes with higher priorities as well as transmissions by downstream nodes producing inversions. $\mathbb{C}_i$ and $|\mathbb{C}_i|$ will symbolize the set of all i downstream nodes with the same *TTRT* as node_*i* and the cardinality of the set, respectively. Obviously, $|\mathbb{C}_i|=k.R(i)+|\mathbb{C}_i| \bmod R(i)$, $k\in\{0, 1, 2...\}$. $|\mathbb{C}_i| \bmod R(i)$ will be notated C_i^m. In what follows it will be assumed that in each case there are enough downstream nodes to produce the upper bound of token inversions.

Lemma 1. In the round starting at $t_0(i)$, up to $R(i)$ nodes may transmit inhibiting the following nodes.

Lemma 2. Following $t_0(i)$, all nodes $\in \mathbb{C}_i$ will transmit before node i.

The number of node_*i*'s placement and token inversions will be notated $p(i)$ and $\tau(i)$ respectively.

Lemma 3. $p(i) = |C_i|$

Lemma 4: Following $t_0(i)$, the number of placement inversions in the last round in which nodes $\in \mathbb{C}_i$ transmit is either $R(i)$ or C_i^m .

The round after $t_0(i)$ in which less than $R(i)$ placement inversions take place will be notated V_0. Topologically, it starts between nodes i and i+1. Successive rounds will be notated V_1, V_2,....

Lemma 5: If $C_i^m=0$, then in V_0 the number of token inversions is $\tilde{R}(i)$.

Lemma 6: If $C_i^m\neq 0$ and $\tilde{R}(i)> C_i^m$, then the number of token inversions in V_0 is $\tilde{R}(i)- C_i^m$.

Lemma 7: If $C_i^m\neq 0$ and $\tilde{R}(i)\leq C_i^m$, then no token inversions take place in V_0.

Lemma 8: The number of placement inversions plus the number of token inversions in V_0, symbolized L_0, is $\max(\tilde{R}(i), C_i^m)$.

Lemma 9: The number of token inversions in V_0 is $L_0 - C_i^m$

By Lemma 9, the total number of inversions suffered by node_*i* in V_0 will be less than $R(i)$ and therefore it would be able to transmit in that round. Messages, however, may be generated in the *i*_upstream nodes in number such that, added to the inversions, inhibit node_*i*. In the next rounds, and although they have heard high priority messages, some *i*_downstream nodes may still be able to transmit, producing new inversions. The process goes on until no more inversions are possible. The round in which this occurs is found in the next theorems.

Theorem 1: If they exist, token inversions take place up to and including the round V_r, where $r = \max x \in N \cup \{0\} \mid x\,(\tilde{R}(i) - R(i)) + L_0 > 0$.

Theorem 2: The total number of token inversions that node_*i* may suffer is

$$\tau(i) = \frac{(r^2 + r).(\tilde{R}(i) - R(i))}{2} + (r+1).\max(\tilde{R}(i), C_i^m) - C_i^m$$

Corollary: The total number of inversions (placement plus token) that node_i may suffer is

$$I(i) = p(i) + \tau(i) = |C_i| + \frac{(r^2 + r)(\tilde{R}(i) - R(i))}{2} + (r+1).\max(\tilde{R}(i), C_i^m) - C_i^m$$

V. Schedulability test

For each node belonging to the lower classes, the expression above provides information necessary to test if the network is schedulable. Each of those nodes must tolerate its own worst case of inversions plus the load coming from nodes with higher priority. Its period must then be

bigger than or equal to the $(I(i)+1)^{th}$ empty slot in the system of $(i\text{-}1)$ nodes. This necessary condition is expressed as

$$\forall i \in \{n_6+1,\ldots,m\} \qquad T_i \geq e_{(I(i)+1)(i-1)} \tag{2}$$

where $e_{(I(i)+1)(i-1)} = \text{least } t | t = 1 + I(i) + \sum_{h=1}^{i-1} \left\lceil \frac{t}{T_h} \right\rceil$

If expressions (1) and (2) hold, the system is RM schedulable.

VI. Conclusions

In general, any method to test the RM schedulability of a LAN needs, as necessary data, the number of priority inversions that each node may suffer. Although timed-token protocols, a taxonomic family to which 802.4 belongs, have been extensively studied, results obtained for other protocols, e.g. FDDI, are by far not directly translatable to it. The presented approach is highly theoretical but practical applications are immediate. In fact, is is very easy to program the final expressions and to carry out the test on the RM schedulability of an 8802/4 network using the empty-slots method. The importance of the result is obvious since, on one hand, 8802/4 has been adopted for the physical layer and the MAC sublayer of the Manufacturing Automation Protocol, and, on the other hand, RMS is a *de facto* RT priority standard supported, amonh others, by the USA Department of Defense.

References

[1] IEEE Std. 802.4. "Token-Passing Bus Access Method and Physical Layer Specifications". *Institute of Electrical and Electronics Engineers, Inc.* 1985.

[2] Janetzky, D. and Watson, K.S.. "Perfomance evaluation of the MAP Token-Bus in real-time applications". *Advances in Local Area Networks*, IEEE Press, New York , pp 411-427. 1987.

[3] Joseph, M. and Pandya, P. "Finding response times in a real time system". *The Computer Journal*, 29,5, pp. 390-395. 1986.

[4] Lehozcky, J. P., Sha, L. and Ding, Y. "The rate monotonic scheduling algorithm: Exact characterization and average case behavior". In *Proc. Real Time Systems Symp.*, IEEE CS, Los Alamitos, CA. 1989.

[5] Liu, C.L. and Layland, J.W. "Scheduling algorithms for multiprogramming in hard real time enviroments". *J. ACM*, 20,1, pp. 46-61. 1973.

[6] Orozco, J. and Santos, J. "Diagramación de tareas en sistemas operativos de tiempo real PMC con inversión de prioridades". In *Proc.* 23 *Jornadas Argentinas de Informática e Investigación Operativa*, pp. 1.25-1.35. 1994.

[7] Rzehak, H., Elnakhal, A. and Jaeger R. "Analysis of Real-Time Properties and Rules for Setting Protocol Parameters of MAP Networks". *The Journal of Real-Time Systems*, 1,3, pp. 221-241. 1989.

[8] Santos, J., Gastaminza, M.L., Orozco, J., Picardi, D. and Alimenti, O. "Priorities and protocols in hard real-time LANs: implementing a crisis-free system". *Computer Communications*, 14,9, pp. 507-514. 1991.

[9] Santos, J., Gastaminza, M.L., Orozco, J. and Matrangolo, C. "802.5 priority machanism in hard real-time RMS applications". *Computer Communications*, 17,6, pp. 439-442. 1994.

Diagramación Dinámica de Tareas de Tiempo Real Duro con Constricciones de Precedencia en Sistemas Distribuidos Multitarea-Multiprocesador

R. Cayssials, J. Orozco, J. Santos y E. Ferro.
Dep. Ingeniería Eléctrica, Instituto de Ciencias e Ingeniería de Computación,
Universidad Nacional del Sur, 8000 Bahía Blanca, Bs. As., Argentina.

E-mail: {iecayss, ieorozco, iesantos, ieferro}@criba.edu.ar
FAX: 00 54 91 553122

Resumen: En este trabajo se presenta un método para la diagramación de tareas apropiativas y cooperativas con constricciones de precedencia en un ambiente distribuido multitarea- multiprocesador. Se establecen las condiciones suficientes para la diagramación del sistema y se obtiene, como consecuencia, el algoritmo que permite analizar su factibilidad.

1. Introducción

El problema de diagramación en sistemas multitarea-monoprocesador es de complejidad polinomial y desde el trabajo liminar de Liu y Layland [1], diversos métodos han sido propuestos para resolverlo con eficacia creciente tanto desde el punto de vista de la utilización del procesador [2, 3, 4, 5] como desde el punto de vista de compartir recursos [6, 7], operando bajo distintas disciplinas de prioridades. Estas están relacionadas, en general, con los vencimientos. De ellas, la más importante, por ser un estándar *de facto* impuesto por el Departamento de Defensa de EEUU, es la denominada de Períodos Monotónicos Crecientes, PMC (*Rate Monotonic Scheduling,* RMS). En sistemas de tiempo real multitarea-multiprocesador los resultados son más escasos, quizá porque, en general, la diagramación de estos sistemas es NP-duro.

Por otro lado, tanto en sistemas mono o multiprocesador, la mayor parte de las investigaciones de la teoría clásica se encuentran en el campo de la diagramación estática donde, el algoritmo de diagramación, posee un completo conocimiento del conjunto de tareas y sus

características tales como vencimiento, tiempo de ejecución, precedencia y futuros tiempos de arribo (instante en que la tarea está lista para ser ejecutada). Sin embargo, pocos resultados se conocen en el campo de la diagramación dinámica [8]. Un algoritmo de diagramación dinámica, en este contexto, posee un conocimiento completo de las *tareas listas* para ser ejecutadas pero no conoce cuando pueden producirse nuevos arribos.

Desde el punto de vista operativo, los algoritmos de diagramación de sistemas multiprocesador pueden tener alcance global o local. En un diagramador global la disciplina de diagramación es aplicada al conjunto total de tareas sobre todos los procesadores. Si es de alcance local, existe un diagramador autónomo en cada procesador que actúa sobre las tareas que le fueron previamente asignadas.

En [1] se ha demostrado formalmente que para un sistema monoprocesador-multitarea, el peor estado de carga es el arribo simultáneo de todas las tareas. Sin embargo, en los casos de sistemas multiprocesador, cuando las tareas cooperan en su ejecución formando parte de un trabajo y son asignadas a procesadores distintos, la verificación local en cada uno de los procesadores puede encontrar el inconveniente de que los tiempos entre arribos no son exactamente conocidos.

Entre pares de tareas puede haber una relación de precedencia en el sentido de que, para comenzar su ejecución, una de ellas (*sucesora*) necesita datos producidos por la otra (*predecesora*). La llegada de dichos datos desde todas las predecesoras al procesador que aloja a la tarea sucesora define (en el contexto de este trabajo) el instante de arribo de la misma. En este caso, los tiempos entre arribos no sólo no son conocidos con exactitud sino que pueden llegar a ser menores que el período mínimo del trabajo al que pertenecen. En consecuencia, no son aplicables directamente las disciplinas de diagramación para sistemas monoprocesador.

En este trabajo, se diseña un mecanismo de diagramación local dinámico para tareas con precedencia en sistemas multiprocesador que permite utilizar resultados desarrollados para sistemas multitarea-monoprocesador. Se establecen también las condiciones de suficiencia para que un sistema sea factible mediante este mecanismo. El método de las ranuras vacías es utilizado como base de análisis de la diagramabilidad del sistema.

Debe hacerse notar que la diagramación dinámica es imprescindible para asegurar la factibilidad de un sistema multitarea-multiprocesador en el cual de ha logrado una asignación

tentativa por alguno de los métodos heurísticos o de simulación conmunmente utilizados [7, 9, 10, 11].

2. Método de las ranuras vacías.

Este método fue presentado en [12] para resolver el problema de diagramación de Redes Locales en tiempo real. Luego fue extendido, en [5] a la diagramación multitarea apropiativa en monoprocesador, aún con granularidad restringida.

El tiempo se considera ranurado y la duración de la ranura es tomada como unidad de tiempo. Las ranuras son denominadas t y numeradas 1, 2, ... Las expresiones *instante t* y *comienzo de ranura t* son equivalentes. Un conjunto de m tareas $\tau_1, \tau_2, \ldots, \tau_m$ asignadas a un procesador queda completamente especificado como $S(m)= \{(C_1, T_1, D_1), (C_2, T_2, D_2), \ldots, (C_m, T_m, D_m)\}$, donde C_i, T_i, D_i representan el máximo tiempo de ejecución, el mínimo tiempo entre arribos y el vencimiento de la tarea τ_i respectivamente.

En [1] se ha demostrado que la disciplina de Períodos Monotónicos Crecientes (PMC), es la mejor entre las de prioridades fijas en el sentido de que si un sistema es diagramable por alguna disciplina de prioridades fijas, también lo será por PMC.

En [5] se ha demostrado que la j-ésima ranura vacía que deja el sistema de m tareas luego del instante de peor estado de carga (arribo simultáneo de todas las tareas en el instante inicial), notada $e_{j(m)}$, es

$$e_{j(m)} = \text{menor}(t) \mid t = j + \sum_{h=1}^{m} C_h \left\lceil \frac{t}{T_h} \right\rceil \qquad (1)$$

En el mismo trabajo, se demuestra que, un sistema $S(m)$ es diagramable bajo PMC si y sólo si para todo $i = 1, 2, 3, .. m$, se cumple $D_i \geq e_{C_i(i-1)}$.

3. Relaciones de Precedencia

Es usual que en Sistemas de Tiempo Real (STR) resulte necesario que las tareas no sean independientes y cooperen entre sí para lograr un determinado objetivo. Cuando una tarea τ_j necesita total o parcialmente los resultados producidos por otra tarea τ_i para poder ser ejecutada, se establece una relación de precedencia, notada $\tau_i \prec \tau_j$ (τ_i precede a τ_j), la cual establece un orden

parcial sobre las tareas. Si $\tau_i \prec \tau_j$ y no existe una $\tau_l | \tau_i \prec \tau_l$ y $\tau_l \prec \tau_j$ entonces τ_i y τ_j son denominadas predecesora y sucesora respectivamente. Este modelo de computación puede ser asociado a un grafo G [11]. Cada nodo del mismo simboliza una tarea y existirá un arco dirigido desde el nodo que representa τ_i al nodo que representa τ_j si y sólo si ellas son predecesora y sucesora respectivamente. G = {Γ,P}, donde Γ es el conjunto de tareas que forman el trabajo y P es el conjunto de sus relaciones de precedencia.

Una relación de precedencia agrega complejidad a la diagramación. En [13] se ha resuelto en forma óptima, con complejidad $O(n^2)$, un problema simple de un conjunto de tareas no apropiativas, idéntico tiempo de arribo y con condiciones de precedencia. Este método, lamentablemente, no es suficiente para la mayoría de los problemas de interés en STR, en los cuales las tareas no tienen un instante conocido de arribo. En [14] se ha demostrado que el problema de diagramación de tareas con diferentes tiempos de arribo y con relación de precedencia es NP-duro. Los mejores resultados sobre precedencia fueron obtenidos a partir del trabajo sobre subclases de la relación de precedencia, alcanzándose algoritmos polinómicos para relaciones *intrees* (un solo sucesor) y *outtrees* (un solo predecesor). Estos casos son incluidos en un grupo más interesante denominado serie-paralelo que es definido en forma recursiva y sobre el cual existen soluciones eficientes, aunque, lamentablemente, los problemas no son comunes en la práctica. Si el sistema es apropiativo, se reduce la complejidad del diagramador y es solucionable en $O(n^2)$ aplicando el algoritmo de Baker [15].

4. El Modelo del sistema

El sistema está compuesto por p procesadores que deben ejecutar n trabajos. El trabajo j está integrado por $m(j)$ tareas, numeradas τ^j_1, τ^j_2,..., $\tau^j_{m(j)}$. El conjunto de n trabajos está completamente especificado por $\mathbb{J} = \{(T_1, D_1, \Lambda_1), (T_2, D_2, \Lambda_2), \ldots, (T_n, D_n, \Lambda_n)\}$ donde T_j, D_j y Λ_j representan el período mínimo, el vencimiento y el conjunto de tareas que componen el trabajo j respectivamente. El conjunto de $m(j)$ tareas que integran el trabajo j está completamente especificado por $\Lambda_j = \{(C^j_1, \Phi^j_1), (C^j_2, \Phi^j_2), \ldots, (C^j_{m(j)}, \Phi^j_{m(j)})\}$ donde C^j_i y Φ^j_i representan el máximo tiempo de ejecución de la tarea i y el conjunto de sus tareas predecesoras respectivamente.

Cada tarea está asignada a un solo procesador. Tareas del mismo trabajo pueden estar asignadas a distintos procesadores. Cada procesador tiene un conjunto de tareas asignadas. En cada procesador existe un diagramador y sólo actúa sobre las tareas asignadas a ese procesador. Esto equivale a decir que el diagramador es local en cada procesador.

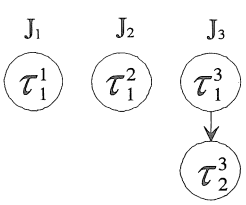
Una tarea puede activarse sólo una vez que hayan terminado de ejecutarse todas sus predecesoras. La terminación de todas las tareas predecesoras produce el arribo de la sucesora. El arribo de las tareas que no poseen predecesoras ($\Phi = \varnothing$), denominadas *raíces*, coincide con el del trabajo al que pertenecen. Las tareas que no poseen sucesora serán denominadas *hojas*.

5. El Mecanismo de Diagramación por Activación Temporizada

5.1. *Ejemplos*

Para ilustrar conceptualmente el mecanismo, se ejemplifica a continuación un mismo conjunto de trabajos con distintas condiciones de arribo y ejecución.

Sean tres trabajos, J_1, J_2 y J_3 y dos procesadores p_1 y p_2. J_1 y J_2 constan de una tarea cada uno (τ^1_1 y τ^2_1). J_3 está formado por dos tareas con relación de precedencia ($\tau^3_1 \prec \tau^3_2$). En las tablas I y II se consignan los parámetros temporales de los trabajos y de las tareas, como así también sus Λ y Φ.



Trabajo	T	D	Λ
J_1	6	6	$\{\tau^1_1\}$
J_2	8	8	$\{\tau^2_1\}$
J_3	8	8	$\{\tau^3_1, \tau^3_2\}$

Tabla I

Tarea	C	Φ
τ^1_1	3	$\varnothing$
τ^2_1	6	$\varnothing$
τ^3_1	3	$\varnothing$
τ^3_2	2	τ^3_1

Tabla II

τ^1_1 y τ^3_1 son asignadas a p_1, en ese orden de prioridad. τ^3_2 y τ^2_1 son asignadas a p_2, en ese orden de prioridad. τ^2_1 arriba en t=7, en sincronismo con τ^3_2 (Fig. 1).

Como puede verse en t=15, τ^2_1 pierde su vencimiento y el sistema no cumple con sus constricciones de tiempo. La no factibilidad proviene del arribo de τ^2_1 en un instante distinto de t=1 con lo que, el tiempo entre arribos de τ^3_2 es 5 en lugar de 8 que corresponde al período del

trabajo J_3. Este simple contraejemplo alcanza para probar que, para sistemas multiprocesador con tareas con relación de precedencia, el peor estado de carga no necesariamente es el arribo simultáneo de todos los trabajos, como sucede en los sistemas estudiados por Liu y Layland.

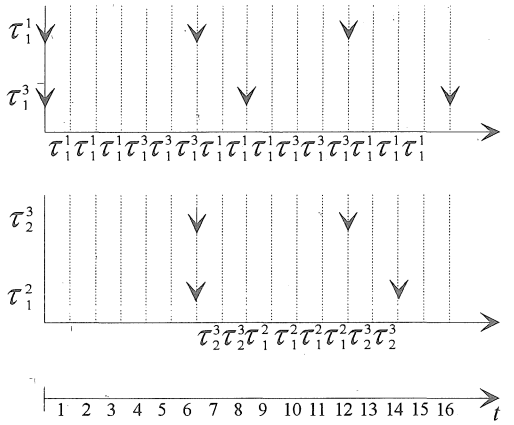


Figura 1

Analicemos ahora otro cronograma de obedecimiento (Fig. 2). En t=12, en lugar de conmutar p_2 a τ^3_2, se sigue ejecutando τ^2_1. En t=14, ésta se concluye y a *posteriori* se ejecuta τ^3_2, cumpliendo ambas con sus constricciones de tiempo.

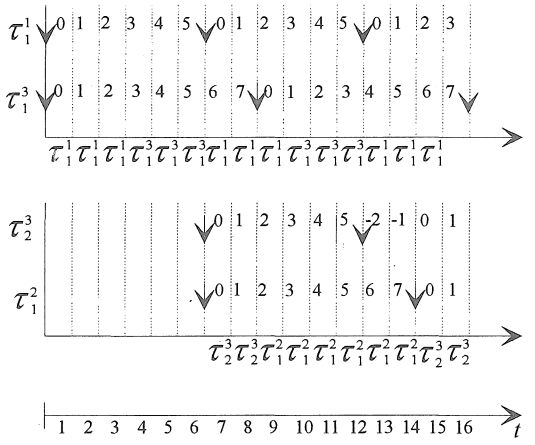


Figura 2

Nótese que, en el ejemplo, si a τ^2_1 se le asigna mayor prioridad que a τ^3_2, a pesar de tener períodos idénticos el sistema no resulta factible. Esto se debe a la relación de precedencia de τ^3_2 con τ^3_1 que adelanta el vencimiento de la primera para poder cumplir con el vencimiento de J_3.

5.2. *El Diagramador*

En lo que sigue, se denominará *Máximo Tiempo de Respuesta* de la τ^j_i, notado MTR^j_i, al máximo número de ranuras que la tarea tardará en finalizar su ejecución. Desde su arribo deberá haber ranuras suficientes para su propia ejecución y para la ejecución de tareas de mayor prioridad.

Consideremos ahora un diagramador local para cada procesador. Además de la tabla de prioridades PMC y la tabla de los MTRs, el diagramador dispone de un contador para cada tarea y opera de acuerdo a las siguientes reglas:

1. Los contadores son incrementados en una unidad en cada ranura.
2. Cada vez que una tarea predecesora termina de ejecutarse pasa a su(s) sucesora(s) un valor que es la diferencia entre el valor de su contador asociado y su MTR+1. Esta diferencia se denomina, *Valor de Inicialización del Contador de la Sucesora,* notado VICS.
3. Las tareas raíz inicializarán el contador en cero cuando se produzca su arribo.
4. La(s) tarea(s) sucesora(s) inicializan su contador con el VICS de la predecesora. Las tareas que posean más de una predecesora tomarán el menor VICS enviado por todas sus predecesoras.
5. El diagramador sólo podrá activar tareas que tengan un valor negativo en el contador si no existe ninguna con valor positivo.

El mecanismo se ilustra en la Fig. 2, donde se muestra la evolución de los contadores de las distintas tareas del ejemplo anterior. Dado que en p_1 se ejecutan τ^1_1 y τ^3_1, resulta $MTR^3_1=6$, valor que corresponde al orden de prioridades τ^1_1 / τ^3_1. En t=7, los contadores de τ^2_1 y τ^3_2 tienen el mismo valor (0). La regla de desempate asigna mayor prioridad a τ^3_2, que, en consecuencia, se ejecuta. En t=9 comienza la ejecución de τ^2_1. En t=12, el valor del contador de τ^2_1 es 6 mientras que el de τ^3_2 es -2. En consecuencia se sigue ejecutando τ^2_1 que logra completarse en t=14, antes de su vencumiento. Como puede apreciarse en el ejemplo, el mecanismo de activación temporizada por contadores produjo el comportamiento deseado.

5.3. *Formalización del método*

A continuación se demuestra formalmente el método descripto, determinando las condiciones suficientes para que el sistema resulte de diagramación factible.

Definición 1: Se denomina *Máximo Tiempo de Arribo* de τ_i^j, notado MTA_i^j, a la máxima cantidad de ranuras que la tarea podrá tener desplazado su arribo con respecto al arribo de la raíz del trabajo.

Obviamente, el MTA de toda tarea-raíz es cero.

Teorema 1

$$MTR_i^j = \text{menor}(t) \mid t = C_i^j + \sum_{\tau_h^k \in L} C_h^k \left\lceil \frac{t}{T_k} \right\rceil \tag{2}$$

donde el conjunto $\mathbb{L}$ es, a su vez, la unión de dos conjuntos. El primero compuesto por todas las tareas de mayor prioridad que la tarea *i* que no pertenecen al trabajo *j*. El segundo está compuesto por las tareas τ_k^j que perteneciendo al trabajo *j* se ejecutan en el mismo procesador, tienen mayor prioridad que la tarea *i* y $MTA_k^j \mid MTA_k^j + MTR_k^j \in (MTA_i^j, MTA_i^j + MTR_i^j]$.

Demostración:

Según los resultados de [1] el peor caso de carga en un sistema monoprocesador es el arribo simultáneo de todas las tareas de mayor prioridad. En [5] se utiliza este resultado para demostrar que si $S(i\text{-}1)$ es el conjunto de tareas con mayor prioridad que la tarea *i*, la máxima cantidad de ranuras que la tarea *i* tardará en ser finalizar su ejecución es:

$$e_{C_i(i-1)} = \text{menor}(t) \mid t = C_i + \sum_{h=1}^{i-1} C_h \left\lceil \frac{t}{T_h} \right\rceil \tag{3}$$

En nuestro caso, todas las tareas que estarán en condiciones de ser ejecutadas son las de los otros trabajos (considerarlas independientes aumenta la latencia de la tarea *i*) , y las del trabajo al que pertenece la tarea *i* y que finalicen su ejecución dentro del intervalo en que se atiende a la tarea *i*. Aunque la tarea no tenga el mismo instante de arribo que la tarea *i*, el resultado no varía debido a que, por tener el mismo período, puede producir un solo requerimiento en todo intervalo si el sistema es de diagramación factible.

Por ese motivo, en la sumatoria se consideran todas las que pertenecen al conjunto $\mathbb{L}$.

Definición 2: Sea el camino λ^j_i definido por un secuencia de tareas τ^j_1, τ^j_2,..., τ^j_i, pertenecientes al trabajo i, tal que: τ^j_1 es tarea raíz y $\tau^j_h \prec \tau^j_{h+1}$ para $1 \leq h < i$.

Teorema 2

El MTA^j_i será:

$$MTA^j_i = \max \sum_{\tau_x \in \lambda^j_h} MTR^j_x \quad \forall\, \lambda^j_h \mid \tau_h \in \Phi^j_i \tag{4}$$

Demostración:

El arribo de una tarea debe esperar que todas sus predecesoras terminen su ejecución. En consecuencia, si se considera el peor tiempo para cada tarea del camino, el arribo de la tarea i tiene como cota más tardía la finalización de la ejecución del camino más lento.

Definición 3: El *tiempo máximo de ejecución del camino* λ^j_i (TEC^j_i) es definido:

$$\forall\, \lambda^j_i \qquad TEC^j_i = \max \sum_{\tau_k \in \lambda^j_i} MTR^j_k \tag{5}$$

Nótese que

$$TEC^j_i = MTA^j_i + MTR^j_i \tag{6}$$

Teorema 3

El trabajo j será factible si para toda tarea hoja i se cumple:

$$D_j \geq \max \{\, TEC^j_i \,\} \; \forall \lambda^j_i \tag{7}$$

Demostración: El trabajo se ejecutará antes de su vencimiento si el camino más lento lo hace.

Corolario

El sistema será factible con el diagramador propuesto si para todos los trabajos del sistema se cumple el Teorema 3

5.4. Algoritmo para análisis de la diagramabilidad

Las condiciones anteriores son de fácil implementación en un algoritmo.

Para todo trabajo:

I) MTA = 0 para toda tarea raíz.

II) Se calculan todos los MTR de las tareas raíces.

III) Se calcula el MTA de todas las tareas a las que se les haya calculado previamente el MTR de sus predecesoras.

IV) Se calculan los MTR de las tareas seleccionadas en III).

V) Se verifica si se incluyeron todas las tareas del trabajo en el conjunto L. Si no se incluyeron, se vuelve a IV). En caso contrario se vuelve a III) hasta concluir con todas las tareas-hoja.

VI) Se calcula el TEC (ec. 6) para todos los caminos que finalicen en una tarea-hoja del trabajo.

VII) Si el máximo TEC de los calculados en VI) es menor que el vencimiento del trabajo, éste será factible.

6. Conclusiones

El trabajo propone una solución para un problema muy poco tratado hasta la actualidad como es la operación de diagramadores dinámicos en sistemas multitarea-multiprocesador. La solución tiene como base el empleo de contadores asociados a cada una de las tareas y cuyo contenido es utilizado por el diagramador para activar o no las tareas arribadas. Se describe así mismo un algoritmo simple para determinar si dada una asignación tentativa obtenida por los métodos heurísticos o de simulación usuales, el sistema es factible.

En trabajos futuros se agregará el estudio de sistemas con variaciones en los relojes de los procesadores y retardos en las comunicaciones entre procesadores. El mecanismo permite estas modificaciones ya que los tiempos de sincronización son relativos al último arribo y no, como en otros sistemas, relativos al arribo inicial. En estos últimos, la integración del error en un período considerable hace que no se pueda acotar el error por sincronización.

Un aspecto importante es la baja complejidad del diagramador: O(n). Se aprovecha las características de prioridades fijas en el sentido de que no existen ranuras vacías si hay requerimientos pendientes.

7. Referencias

1. Liu, C.L. and Layland, J.W., "Scheduling algorithms for multiprograming in hard real time enviroments", *J. ACM*, Vol. 20, N° 1, pág. 46-61, 1973.
2. Joseph, M. and Pandya, P., "Finding Response Times in a Real-Time System", *The Computer Journal*, Vol. 29, N° 5:390-395, 1986.
3. Leung J., Whitehead J., "On the complexity of Fixed Priority Sheduling of periodic, Real -Time tasks", *Perfomance Evaluation*, Vol. 2, N° 4, pág. 237-250, 1982.
4. Lehozcky, J.P., Sha, L. and Ding, Y., "The Rate Monotonic Scheduling Algorithm: Exact characterization and average case behavior", In *Proc. Real Time Symp. IEEE CS*, Los Alamitos, CA. 1989.
5. Santos, J. and Orozco, J., "Rate Monotonic Scheduling in Hard Real-Time Systems", *Information Processing Letters*, N° 48, pág. 39-45, 1993.
6. Sha L., Rajkumar R., Lehoczky J.P., "Priority Inheritance Protocols: An aproach to Real-Time Synchronization", *IEEE Trans. Computers*, Vol. 39, N° 9,pág 1175-1185, 1990.
7. Zhao W., Ramamritham K., Stankovic J., "Preemptive Scheduling under Time and Resource Constraints", special issue on *Real-Time Systems, IEEE Trans. Computers*, Vol. 36, N° 8, pág. 949-960, 1987.
8. Stankovic J.A., Spuri M., Di Natale M., Buttazzo G.C., "Implications of Classical Scheduling Results for Real-Time Systems", *IEEE Computer* , N° 6, Vol. 28, pág 16-25, 1995.
9. Ramamritham K., Stankovic J., Shiah, "Efficient Scheduling Algorithms for Real-Time Multiprocessors Systems", *IEEE Trans. Parallel and Distr. Computting*, Vol. 4, N° 4, pág. 382-397, 1993.
10. Santos J., Ferro E., "An Heuristic Aproach to Allocating and Scheduling NP-Hard Real-Time Systems", *XX Conferencia latinoamericana de Informática,* pág. 739-746, 1994.
11. Tindell K., Burns A., Wellings A., "Allocating hard Real-Time tasks: An NP-hard problem made easy", *Real Time Systems,* pág. 145-166, 1992.
12. Santos J., Orozco J., Alimenti O., "Perfomance Evaluation of Standard Lan Protocols in Time Constrained enviroments", *IEEE INFOCOM'89*. Waterloo, Ontario, Canadá, 1989.

13.Lawler, "Optimal Sequencing of a Single Machine Subject to Precedence Constraints", *Management Science*, N° 19, 1973.

14.Lenstra J.K., Rinnooy Kan A.H.G., "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey", *Ann. Discrete Math*, N° 5, pág 287-326, 1977.

15.Ramamritham, K., "Allocation and scheduling of complex periodic task", *In Proc. 10th International Conference on Distributed Computing Systems*, pp. 108-115, 1990.

EL IMPACTO DE LA TECNOLOGIA DE INFORMACION EN LA EMPRESA DE LOS 90S

Por : Francisco Rueda
Ingeniero de Sistemas y Computación, Universidad de Los Andes
DEA en Informática, Universidad de Grenoble
Profesor Investigador, Departamento de Ingenieria de Sistemas, Universidad de los Andes, Bogotá
Apartado Aéreo 4976 Bogot-a - Colombia
Fax : 2840796 (Bogotá)
Dirección electrónica : frueda@cdcnet.uniandes.edu.co

RESUMEN

En el mundo actual se oye hablar mucho de tecnologías como robótica, EDI, groupware, correo electrónico, etc. y se dice que ellas están transformando a las organizaciones. En este artículo, que fue elaborado a partir de un estudio bibliográfico sobre el tema, se presentan algunos de los cambios más importantes que ellas están induciendo, haciendo énfasis en dos aspectos : la competitividad y la estructura administrativa y organizacional. Es la opinión del autor que el conocimiento de las empresas sobre estos temas es condición indispensable para poder competir eficientemente en el turbulento mundo de la década de lo 90s.

Introducción

Según Bradley et al (1) , los propulsores más destacados del cambio estratégico en el mundo actual son la globalización y la innovación tecnológica, y están acelerándose a un ritmo que los hará aún más importantes en la década venidera.

En opinión de los autores anteriores está ocurriendo una nueva fusión de la tecnología de información y las telecomunicaciones, que afectará radicalmente a todas las compañías, aún a las que no han sido usuarias importantes de tecnología. Esta nueva fusión es extraordinariamente dinámica y va a conducir a cambios fundamentales en la estructura de las firmas, quienes estarán más interconectadas, lo cual favorecerá el trabajo cooperativo y requerirá formas distintas de administración y control . Otro efecto importante de esta fusión consiste en que las estrategias de competitividad de las firmas están siendo y serán crecientemente afectadas por la creación de nuevas industrias, la reestrucutración de otras y el énfasis en la búsqueda de ventajas competitivas.

En este artículo se presentan algunos de los gradientes de cambio más destacados.

El impacto de la Informática en la empresa de los 90s

Las principales conclusiones de un estudio realizado por varios investigadores de la escuela de negocios de la Universidad de MIT con respecto al impacto de la tecnología de información en la empresa de los 90s son los siguientes (3) :

i. Está posibilitando cambios fundamentales en la forma de hacer el trabajo

El grado en el cual un trabajo cambia propulsado por la tecnología de información depende de qué tanto se basa en información. En algunas empresas un porcentaje muy alto de los empleados tienen trabajos con estas características, y existe una tendencia a que esto se refuerce, según Naisbitt (4)

Los impactos sobre el trabajo van a llegar a todos los niveles: el de producción (en donde la robótica va a afectar la producción física, el procesamiento de datos la producción de información como facturación y contabilidad y los sistemas de diseño, herramientas CASE y similares la producción de conocimiento), el de coordinación, por la posibilidad que ofrece esta tecnología de reducir el tiempo y la distancia a cero, y de constituir una memoria organizacional, y el de administración por sus efectos sobre la dirección y el control.

ii. Está posibilitando la integración de funciones del negocio a todos los niveles , dentro de y entre las organizaciones

Una de las razones más importantes del cambio inducido por la tecnología informática tiene que ver con la posibilidad contar con cualquier información, en cualquier momento, de cualquier lugar, y de cualquier manera que se quiera mirar. Esto permitirá la integración de funciones dentro de la cadena de valor en una compañía (al permitir, por ejemplo la integración de el diseño, la ingeniería y la producción) o entre compañías (cuando distintas actividades de la cadena de valor se pueden realizar en diferentes empresas) y abrirá posibilidades como mercados electrónicos, que involucran eventualmente varias organizaciones (los sistemas manejados por las agencias de viajes constituyen un muy buen ejemplo de esto)..

iii. Está causando desplazamientos en el clima de competitividad de las empresas

Uno de los aspectos que más se ha visto afectado por la tecnología informática es el de la competitividad entre las empresas, debido fundamentalmente a las nuevas posibilidades que ofrece a éstas lo cual genera nuevas formas de competir y colaborar. La creación de nuevas empresas y la alteración de las condiciones de interrelación con clientes, proveedores y competidores son quizás las razones más importantes para la ocurrencia de este fenómeno.

iv. Presenta nuevas oportunidades estratégicas para que las organizaciones reevaluen su misión y sus operaciones

La experiencia muestra que la introducción de la tecnología informática puede contribuir nó sólo a reevaluar las condiciones de competitividad de una empresa sino también a redefinir su misión y sus operaciones. En opinión de Scott (3) esto se logra a través de tres etapas, la automatización (que permite reducir los costos de producción) , la informatización (que se logra cuando la automatización genera información adicional que puede ser usada en provecho de la empresa) y la transformación (que es un estado caracterizado por liderazgo, visión y un proceso sostenido de empoderamiento de la organización).

v. La aplicación exitosa de la tecnología de información requerirá cambios en la estructura administrativa y organizacional

El uso de tecnología informática puede contribuir a la redefinición de la organización, entre otras razones por permitir la distribución de poder, función y control, para dirigirlo hacia donde es más efectivo, teniendo en cuenta la misión y los objetivos de la organización y su cultura.

Las organizaciones han manejado tradicionalmente alguna forma de estructura matricial, que involucra productos, mercados, funciones y geografía. Con la disminución de los costos de coordinación estas formas de organización deben ser repensadas. Por ejemplo la posibilidad de constituir grupos de trabajo multigeográficos abre nuevas posibilidades que conllevan a cambios en las formas tradicionales de medición, recompensas, incentivos y cualidades requeridas.

vi. Un desafío muy importante para la administración de los 90s consistirá en conducir a sus organizaciones a través de las transformaciones necesarias para prosperar en un ambiente competido globalmente.

Las conclusiones del estudio de MIT, reseñadas más arriba, nos muestran un panorama muy amplio con respecta a la influencia de la tecnología de información en las empresas. En las siguientes secciones centraremos la atención en dos aspectos : el impacto en la competitividad y en la estructura administrativa y organizacional.

La tecnología de información y la competitividad de las empresas

Según Bradley et al (1) la tecnología inducirá la creación de muchas industrias nuevas y, más importante aún, la reestructuración de muchas de las existentes, entendiendo por ésta el cambio de las bases de ventajas competitivas de las industrias y la modificación de las maneras en las cuales los proveedores, los clientes y las empresas rivales compiten, cooperan y en general interactúan entre sí.

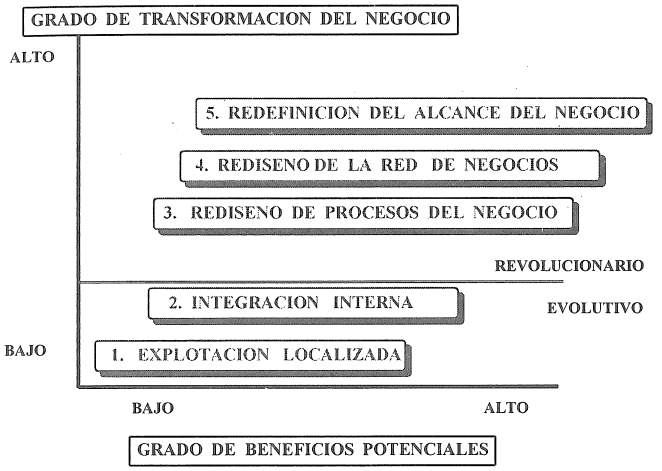
Podrían darse muchos ejemplos de la afirmación anterior, pero para nuestros propósitos basta con citar uno : lo ocurrido en la bolsa de valores de Londres, la cual tuvo importantes transformaciones impulsada principalmente por dos factores : las políticas de desregulación y la introducción de nuevas tecnologías. Las primeras permitieron una mucha participación de firmas en el mercado y terminaron con las comisiones fijas. La segunda aumentó substancialmente la eficiencia del mercado permitiendo el acceso a la información a todos los interesados, en forma independiente de su localización física. Los cambios inducidos fueron dramáticos : todos los comisionistas se instalaron en sus oficinas en donde pudieron contar con más facilidades para su trabajo, como recursos de información, modelos de computador y clientes; la participación extranjera llegó a constituir cerca del 50% , mientras en otras bolsas esta apenas alcanza el 2% o 5 %, lo cual hizo que muchos negocios se desplazaran hacia Londres. Además de lo anterior los márgenes de las empresas comisionistas se redujeron substancialmente, conduciendo en algunos casos a importantes pérdidas.

Mostraremos a continuación diferentes puntos de vista que muestran cómo puede la tecnología informática inducir cambios en la estrategia de la empresa.

Diferentes niveles de impacto de la tecnología de información en la estrategia

Venkatraman (5) dice que las empresas deben redefinir la función de la tecnología de información, desde su posición clásica de soporte, hasta una en la cual pueda jugar un papel crítico en la formulación y en la implementación de la estrategia

Según el autor mencionado la reconfiguración del negocio teniendo en cuenta la tecnología de información puede tener varios grados como se ilustran en la siguiente gráfica.



Como puede verse, el grado de beneficios potenciales depende del grado de transformación, el cual puede ser de cinco tipos, descritos a continuación.

La *explotación localizada* se refiere al uso de la tecnología de información dentro de las funciones del negocio, como manufactura o mercadeo, o aún a actividades de negocios aisladas dentro de las funciones. Con lo anterior se obtienen mejoras en la eficiencia sin impactar áreas organizacionales relacionadas.

Como ejemplos de aplicaciones con estas características son los sistemas CAD/CAM, los sistemas de soporte al cliente y los sistemas de entrada de pedidos.

La *integracion interna* se refiere a la explotación de la tecnología de información en diferentes actividades en los procesos del negocio. Hay dos tipos importantes de integración : técnica y organizacional. La primera se refiere a los equipos y la segunda a las actividades.

Como ejemplo de una aplicaciones con estas caracterísitcas se puede mencionar el sistema que soporta la cuenta de manejo de dinero en la compañía Merril Lynch que inició un proceso que acercó a las compañías de servicios financieros con los bancos. Esta es una cuenta consolidada de manejo de activos que puede incluir cuentas de cheques y de inversión , tarjetas de crédito y débito, mercado de dinero y fondos mutuos. Esta cuenta tiene la característica de presentar al cliente en forma unificada una línea completa de servicios de la compañía, integrando y combinando lo que antes eran líneas separadas de negocios. Los cambios inducidos por este sistema fueron dramáticos, pues le permitieron a Merril Lynch conservar sus clientes aún ante la existencia de ciclos financieros que hacen que algunas inversiones dejen de ser atractivas y que otras surjan, al poder ofrecer una gama amplia de alternativas dentro de la cuenta y generando barreras psicológicas que evitan que los clientes vayan a otra parte.

El *rediseño de procesos del negocio* se refiere a su reconfiguración, usando la tecnología de información como punto central. Refleja un esfuerzo conciente de alinear la infraestructura de tecnología de información con los procesos del negocio, en lugar de simplemente sobreponer la plataforma tecnológica a estos.

Un ejemplo de lo anterior es Batterymarch, una compañía de manejo de fondos de pensiones que pudo reducir substancialmente sus necesidades de personal trabajando con acciones no apetecidas por manejadores de portafolios, logrando así una mayor eficiencia a través del rediseño del negocio.

El *rediseño de la red de negocios* se refiere a la reconfiguración del alcance y de las tareas de la red de negocios involucrada en la creación y suministro de bienes y servicios. Incluye las tareas dentro y fuera de los límites formales de la organización, y constituye una alternativa a las tradicionales integración horizontal y vertical. Es importante aclarar que este concepto es diferente al de EDI que tiene que ver con los aspectos técnicos de las redes y el intercambio de información entre compañías.

Un ejemplo de interconexión entre compañías es el de las redes interbancarias como Servibanca o Redeban que permiten compartir infraestructura tecnológica. Otro ejemplo son los sistemas de reservas de aerolíneas como el Sabre de American Airlines que integran varias compañías. Sin embargo, el verdadero rediseño de la red de negocios se logra cuando cada compañía desempeña un rol específico en el sistema de información, en un mercado especializado, lo cual crea oportunidades para explotar creativamente las capacidades de la tecnología de información.

La *redefinición en el alcance del negocio* se refiere a la posibilidad de ampliar la misión y/o el alcance del negocio, a través de productos y servicios relacionados.

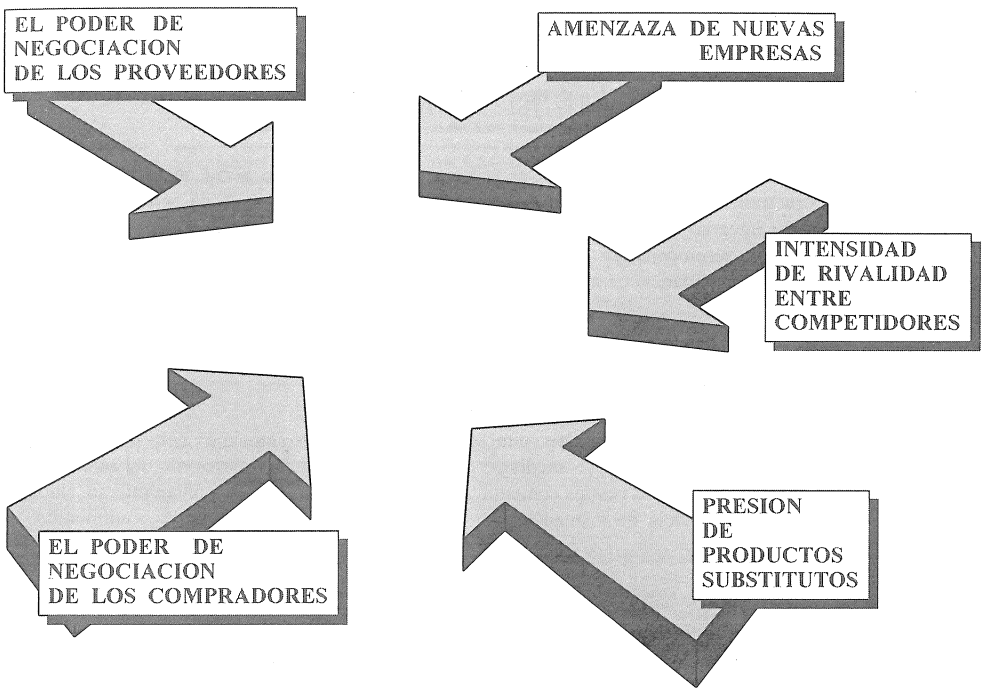
Un ejemplo es el sistema Otisline de la compañía Otis, que soporta el proceso de mantenimiento de sus equipos (ascensores, escaleras mecánicas,...) en todo el país mediante un sistema en línea. Lo interesante del asunto no es sólo que la compañía haya podido aumentar su ventaja competitiva al mejorar su servicio de mantenimiento, sino que le abrió un nuevo mercado para el servicio de mantenimiento de ascensores de otras compañías, ampliando el dominio del negocio.

Algo parecido podemos decir del sistema Sabre, de American Airlines, que no sólo le permitió a la compañía prestar un mejor servicio, sino que además hizo posible que otras compañías lo usaran, generando con esto importantes ganancias, lo cual hizo que esta actividad se convirtiera en una de sus actividades importantes..

El papel de la tecnología en la competitividad de las empresas

Según Bradley et al (1) cuando las firmas invierten en tecnología para obtener ventajas competitivas están tratando de disminuir sus costos, aumentar el desempeño de sus productos y servicios a los ojos del consumidor o mejorar la velocidad de respuesta. Las ventajas buscadas no son absolutas sino relacionadas con la competencia.. Lo que se busca es dismunir los costos por medio de economías de escala o alcance, reducir del costo de las transacciones, explotar los vínculos entre varias actividades de la compañía o administrar la interrelación con otros negocios de la firma. También puede buscarse aumentar la diferenciación respondiendo más efectivamente a las necesidades del comprador con respecto a la calidad, desempeño, servicio, diferentes aspectos o ciclos de tiempo

Una forma de entender el impacto de la tecnología informática en la competitividad, en opinión de Ives (9), es tener en cuenta los factores que inciden en la competitividad de las empresas, según Porter, los cuales se muestran en la siguiente figura :



Según el autor mencionado, la tecnología de información puede tener un impacto importante en cada uno de los elementos mostrados en la figura : la amenaza de nuevas empresas, la intensidad de la rivalidad entre los competidores, la presión de productos substitutos, el poder de negociación de los compradores y el poder de negociación de los proveedores.

Ya vimos que una de las posibilidades que ofrece la tecnología de información es la*creación de nuevas empresas,* basadas en ella. Los servicios de información de crédito o los servicios de valor agregado de diferente tipo constituyen ejemplos al respecto. Lo interesante de este tipo de nuevas compañías, en nuestro caso, es que pueden afectar las condiciones de competitividad del medio.

El ejemplo de la bolsa de Londres dado más arriba sirve para ilustrar cómo la tecnología informática puede contribuir substancialmente a *modificar la intensidad de rivalidad entre competidores.*

La tecnología pude contribuir a crear *productos substitutos* como es el caso de los sistemas EDI que de alguna manera constituyen una alternativa a la la distribución de encomiendas o a crear barreras de entrada para ellos, como en el caso de los sistemas de reservas de las aerolíneas que han hecho muy difícil que otras empresas entren a prestar servicios similares teniendo en cuenta la dificultad de replicar su infraestructura tecnológica y su "know how" .

En el mundo de la computación la tecnología de sistemas propietarios, ya en vías de extinción por el surgimiento de los estándares, nos proporciona un muy buen ejemplo de cómo se puede reducir severamente la*capacidad de negociación de los clientes.* Otro ejemplo a la mano es el de la compañía Merril Lynch, mencionado más arriba, que pudo aumentar su capacidad de negociación con los clientes ofreciéndoles un paquete integrado de servicios que hizo que para estos fuera más difícil buscar a empresas de la competencia para la obtención de eso servicios, teniendo en

cuenta la facilidad que entrañaba el poder obtener una gama amplia de servicios financieros a través de una sola empresa.

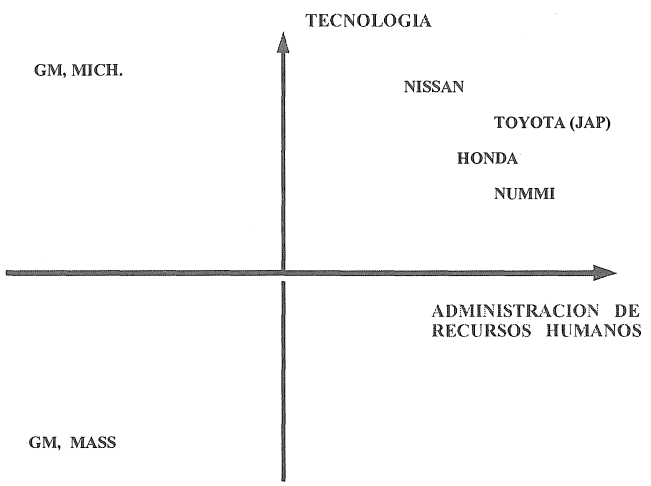
El *poder de negocicación con los proveedores* suministrándoles información estratégica como es el caso de algunos almacenes de cadena que pueden analizar las características de sus clientes y su tendencias de consumo y "realimentar"con esta información a sus proveedores para que estos tomen decisiones de producción, o las que ofrecen servicios de EDI para interconexión con sus proveedores pues, una vez estos han entrado en ellos es muy difícil salirse por las desventajas que esto podría representar desde diferentes perspectivas.

La relación entre la tecnología, la estructura organizacional, los procesos administrativos y los individuos y roles.

Según lo dicho anteriormente la tecnología de información producirá profundos cambios en el ambiente empresarial. Sin embargo, hay que tener presente que ésta por sí sola no garantiza su éxito, sino que se necesita un proceso que involucre a la gente y al cambio organizacional. En lo que sigue analizaremos algunos de los impactos más destacados de la tecnología en diversos aspectos como la estructura organizacional, los procesos administrativos y los individuos y roles.

La relación entre el impacto de la tecnología y las políticas de administración de los recursos humanos

Osterman (10) sostiene que el impacto de la tecnología en una organización depende de la estrategia de manejo de los recursos humanos. Para ilustrarlo nos muestra el caso del sector automotriz en donde se ve claramente este hecho. La siguiente figura ilustra sobre los grados de desarrollo tecnológico y de desarrollo de las políticas de recursos humanos en diferentes empresas de ese sector :



Teniendo en cuenta las características anteriores podemos ahora analizar la productividad y número de defectos en cada una de esas empresas :

	Productiv. (2)	Calidad (1)	Nivel Automat.
Honda (EU)	19.2	72.0	77.0
Nissan (EU)	24.5	70.0	89.2
NUMMI	19.0	69.0	62.8
Toyota (J)	15.6	63.0	79.6
GM. Mich	33.7	137.0	100.0
GM, Mass	34.2	116.0	7.3

(1) **Defectos / 100 unidades**
(2) **Horas requeridas para producir una unidad**

Puede verse que empresas con un alto desarrollo tecnológico pero con un bajo desarrollo en su política de administración de recursos humanos como GM, Michigan tienen resultados de productividad muy bajos, comparables a los de empresas con un bajo desarrollo tecnológico y con políticas pobres de desarrollo de recursos humanos como GM, Mass, mientras empresas como NUMMI, con un alto desarrollo de las políticas de desarrollo de los recursos humanos y un desarrollo tecnológico mediano tienen resultados bastante más favorables.
En el mundo informático no conocemos estudios que muestren una relación tan contundente entre el impacto de la tecnología y el manejo de recursos humanos, pero podemos intuir que las cosas tienen una tendencia similar. Por ejemplo, en las empresas en las que no se han creado las condiciones adecuadas para que los empleados tengan una formación adecuada, y además flexibilidad y creatividad en el desempeño de sus labores podemos esperar que inversiones cuantiosas en tecnologías como el correo electónico, el EDI o el groupware, no se vean recompensadas por aumentos en la eficiencia, la competitividad o el servicio al cliente. Esto está muy relacionado con el denominado nivel de "empoderamiento" de los empleados y su relación con su productividad.

La influencia de la tecnología en los individuos y roles

En cuanto a la relación entre la tecnología y el nivel de empleo Osterman concluye que hay que tener en cuenta tres aspectos : por un lado, podrían ocurrir desplazamientos en el empleo dentro de una empresa, pero estos podrían ser absorbidos por el sistema económico, por otro, los desplazamientos en la empresa dependen de la facilidad que tengan los trabajadores para reubicarse, lo cual nos lleva a concluir que en últimas, todo depende de las características de la empresa.

En cuanto a la relación entre la introducción de la tecnología y la eventual descalificación que esto puede generar en algunos de sus empleados, Osterman afirma que todo depende de la cultura de la empresa, porque si bien puede usar un enfoque taylorista, que proclama que la máquina puede usarse para substituir al trabajador, también puede usarse el de potenciar a éste, "dándole inteligencia a la máquina".

Osterman también opina que hay tres características de la tecnología de información que nos ayudan a entender su impacto en la organización : por una lado, conduce a una integración más estrecha entre las funciones y una mayor interdependencia de las actividades, por otro exige una respuesta más rápida y respuestas en tiempo real, y por

último, las consecuancias de los errores y las "caídas" del sistema son más costosas (en los sistemas de cajeros automáticos que hacen actualizaciones en línea los errores son mucho más peligrosos que cuando todas las actividades se realizaban en forma manual).

Como consecuancia de lo anterior la naturaleza de las habilidades requeridas cambia, desde las manuales hacia las de manejo de datos. Además es muy importante el nivel de responsabilidad, pues los errores cuestan caro.

Además de lo anterior, al existir una tendencia a romper barreras intra e interinstitucionales, se generan varios efectos : necesidad de cambios en los roles de los empleados y administradores al fluir más información hacia y desde los empleados, cambios en la percepción de los supervisores con respecto a su trabajo, al contar con más información pero menos poder sobre sus subordinados, desplazamientos en las carreras de los empleados, al existir una posibilidad de desplazamiento hacia otras dependencias de la empresa o hacia otras empresas.

Según un ejecutivo citado en un artículo " en una empresa inerconectada, el estilo de supervisión cambia. Hay menos supervisión del contenido del trabajo y más del desempeño global de la persona y de su carrera". Lo cual se ve reforzado por la opinión de otro, quien conceptua que " (en la empresa interconectada) tengo que confiar en el profesionalismo y dedicación de mi gente y no en la supervisión directa" (11)

La informática como tecnología coordinadora y de integración

Según Rockart (12) la tecnología de información debe ser vista como un factor de diseño en el cambio organizacional y en la innovación y no sólo como un posibilitador de eficiencia en la organización una vez que se ha hecho un diseño organizacional. Esto permite crear firmas que pueden crecer simultáneamente en tamaño, complejidad y tiempo de respuesta.

En opinión del autor anterior una de las principales características de la tecnología de información es su carácter integrador, el cual puede ser manifestarse en la cadena de valor, dentro de una función, como soporte a grupos de trabajo y en sistemas de control y planeación internos. Lo anterior puede tener profundos impactos en la estructura interna (estructuras administrativas, roles y procesos), en los grupos de trabajo y en la "desintegración" de la empresa, que se manifiesta en darle más énfasis a los mercados y menos a las jerarquías.

Con respecto a la organización de sistemas de información implica cambios pues se requiere una mucho mayor ingerencia de los directivos en las políticas de informática y de los ejecutivos de informática en las polítcas de la compañía, y una preparación mayor de estos últimos. El presidente de una importante compañía se refería al punto anterior diciendo que "... los presidentes de empresa están reconociendo cada vez más el impactro que las decisiones de tecnología tienen en su negocio y en la cultira corporativa. Como resultado, cada vez se sienten menos cómodos delegando las decisiones tecnológicas a otros" (13)

La forma predominante de organización en la empresa tradicional, según Malone y Rockart (14) , es la jerarquía funcional a través de divisiones, la cual se caracteriza por un conjunto de principios de administración centrados alrededor de la noción de jerarquía : cadena de mando, límites de control, comunicaciones basadas en memorandos escritos etc.. Esta forma de organización está poco a poco cediendo el puesto a una alterna, basada en una comunicación punto a punto soportada por la tecnología de información, equipos de trabajo y alianzas estratégicas. Con esta última estructura se puede utilizar mucho más eficientemente la tecnología de información. Malone y Rockart han acuñado el término "tecnología coordinadora" para referirse a este nuevo fenómeno, que es mucho más que la integración de la Informática y las Comunicaciones, pues se refiere a cómo permitir que la gente coopere más efectiva y eficientemente en su trabajo, potenciando enormemente actividades como la comunicación, la toma de decisiones y la asignación de recursos. En opinión de los autores es en esas actividades fuertemente basadas en información que la tecnología informática puede tener sus más profundos impactos. Esto implicaría cambiar la concepción tradicional de los computadores para que dejen de ser máquinas para "calcular" y se conviertan en "tecnologías coordinadoras".

Según los autores mencionados así como en la revolución industrial las firmas triunfadoras fueron aquellas que supieron sacar provecho de la producción en masa y de las tecnologías de transporte, en la revolución informática las firmas que más se beneficiarán serán aquellas serán aquellas que puedan aprovechar de las nuevas tecnologías de coordinación, para poder integrar mejor el trabajo de la gente dentro de las compañías e interconectar más eficientemente las compañías entre sí.

Un fenómeno que puede surgir de la tendencia anterior se refiere al resurgimiento de las organizaciones pequeñas de la era preindustrial, las cuales estarían ahora interconectadas, constituyendo organizaciones virtuales, para resolver un problema o producir un producto en pocos días.

La disminución en los costos de coordinación, según Rockart induce tres efectos importantes en las empresas: *substitución*, *nuevos usos* y *nuevas estructuras*.

Un efecto de primer orden de la reducción de los costos de coordinación, inducido por la tecnología informática, es la *substitución* de la coordinación humana por la electrónica. Por ejemplo podría ocurrir que desaparecieran los mandos medios pues las tareas de comunicación que ellos realizan pueden ser desarrolladas en forma menos costosa por medio de los computadores, lo cual conduciría al aplanamiento de las jerarquías. En lo que tiene que ver con la coordinación entre compañías el efecto podría ser la desaparición o modificación substancial de trabajos como el de corredores de bolsa, empresas de turismo o especialistas de comercio, cuyo principal papel hoy en día es el de servir de intermediarios de información.

Un efecto de segundo orden de reducir los costos de coordinación es el aumento que puede gestarse en ésta. Un buen ejemplo de este fenómeno se presenta con los sistemas de reservas de aerolíneas como American Airlines y United Airlines, los cuales no sólo les han permitido a éstas beneficiarse de las tarifas que cobran por el servicio, sino también han posibilitado cosas como la consideración de más alternativas para los clientes, la definición de tarifas especiales y el ajuste permanente de precios. Algo similar ha ocurrido con un sistema de soporte de mantenimiento utlizado por Otis que le ha permitido aumentar sensiblemente su nivel de coordinación.

Un efecto de tercer orden de la disminución de los costos de coordinación se refiere a un desplazamiento hacia el uso de estructuras más intensivas en coordinación. Por ejemplo la compañía Frito Lay, que cuenta con 10.000 vendedores, le proporciona a estos computadores portátiles en los que registran sus ventas que luego trasmiten a un computador central, el cual a su vez comunica a estos información sobre cambios de precios y promociones. Esto le ha permitido a la empresa descentralizarse y a los directores regionales contar con información sobre las ventas y sugerir modificaciones a las políticas de la compañía.

El aumento en la capacidad de coordinación puede inducir no sólo cambios importantes en las estructuras de la empresa sino también en su relación con clientes, proveedores y competidores. Por ejemplo, los miembros de la Rosenbluth Internacional Alliance, que es un consorcio de viajes alrededor del mundo, comparten los registros de los clientes, servicios y software y proporcionan líneas deayuda gratis en Inglés en los principales países.

Otro efecto importante de la disminución de los costos de coordinación tiene que ver con un aumento en el uso de mercados externos por parte de las compañías, o una reducción en la integración vertical, lo cual conduce a una disminución del tamaño de las compañías. Esto ocurre pues los costos de coordinación con compañías proveedoras se disminuyen substancialmente lo cual hace que se puedan aprovechar las ventajas de escala que pueden obtener estas con mercados más grandes. Una importante implicación de ese argumento es que la tecnología de información hace que los mercados sean más eficientes, produciendo efectos devastadores en aquellas firmas que basan su ventaja estratégica en las ineficiencias del mercado, y favoreciendo aquellos mercados con compradores y proveedores dispersos, diferencias grandes de precios y disponibilidad cambiante de productos.

Adicionalmente a lo anterior, la tecnología de computación , según Malone et. al. (15), conduce a la expansión de los mercados abiertos (con muchos proveedores) por disminuir los costos de coordinación y facilitar la creación de mercados electrónicos que permiten manejar productos especializados y/o cuya descripción es compleja, como es el caso de los servicios turísticos. En este tipo de mercados el comprador puede aumentar el número de alternativas

que pueden ser consideradas y la calidad de la alternativa seleccionada y disminuir el costo de la selección del producto. Los anteriores son ejemplos que ilustran, según los autores mencionados, que el mundo del mañana, interconectado electónicamente, no sólo será más rápido y eficiente sino que habrá cambios fundamentales en la manera como las firmas y los mercados organizan el flujo de bienes y servicios,

Otra transformación importante que pueden generar las tecnologías coordinadoras es el surgimiento de las "adhocracias", que se refieren al uso de equipos de trabajo que se conforman dinámicamente, según las necesidades de un proyecto, y redes altamente descentralizadas de grupos empresariales relativamente autónomos. Las tecnologías informáticas pueden, por ejemplo, ser usadas para encontrar y coordinar gente con conocimientos y habilidades diversos, en diferentes partes de la organización. Esto puede ser enormemente potenciado por la facilidad de estas para hacer más veloz el "metabolismo de información" de las organizaciones, o sea, la tasa a la cual estas capturan, mueven, digieren y responden la información, y por el aumento en la capacidad de compartir la información entre la gente.

Otro efecto importante de las tecnologías de coordinación es que ellas facilitan el surgimiento de organizaciones verdaderamente globales, entendiendo por esto aquellas que incorporan grupos de trabajo de diferentes sitios para el diseño de productos o de nuevas estrategias, o las que tienen mercados de alcance global y no local.

La empresa interconectada permite un mejor desempeño a todos los niveles : dado que la información puede ser distribuida más fácilmente, la gente de los niveles bajos de la organización puede estar mejor informada para poder tomar decisiones más efectivamente, y en los niveles altos se pueden revisar más facilmente las decisiones tomadas en los niveles inferiores. Esto hace que cada uno tenga una sensación de mayor poder.

Como consecuencia de lo anterior hay muchas formas novedosas de utilizar las tecnologías coordinadoras para generar estructuras organizacionales novedosas como redes de consultoría electrónica, redes de decisión mediatizadas por tecnología informática, etc.

Conclusiones

La tecnología de información está generando profundos cambios en las organizaciones a una velocidad tan vertiginosa que muchas de ellas no han tenido tiempo para captarlos y asimilarlos. La amplitud de estos cambios se puede apreciar muy bien en las conclusiones del estudio de MIT : está posibilitando cambios fundamentales en la forma de hacer el trabajo, está posibilitando la integración de funciones del negocio a todos los niveles , dentro de y entre las organizaciones, está causando desplazamientos en el clima de competitividad de las empresas y presenta nuevas oportunidades estratégicas para que las organizaciones revaluen su misión y sus operaciones. Sin embargo, para que lo anterior pueda ser realizado exitosamente, nos advierte el mismo estudio, se requieren cambios en la estructura administrativa y organizacional.

En este artículo se presentan algunos de los cambios más destacados que, de acuerdo con la literatura, está induciendo la tecnología computacional en dos aspectos : la estrategia y la estructura administrativa y organizacional.

El futuro de las empresas en los años venideros dependerá en gran medida de la capacidad que tengan para asimilar los cambios que está induciendo la tecnología informática no sólo en su estructura administrativa y organizacional sino también en la forma de competir. Espero que las reflexiones contenidas en este artículo puedan servir para alimentar este proceso.

Bibliografía

(1) BRADLEY S. P. , HAUSMAN J. A. , NOLAN R. L. , Global competition and technology. En S. P. BRADLEY , J. A. HAUSMAN, R. NOLAN (editores) (1993), *Globalization,Technology, and Competition.* Boston Massachusetts. Harvard Business School.

(2) YATES J. y BENJAMIN . The past and present as a window on the future . En M. S. SCOTT (1990) *The Corporation of the 1990s* . New York Oxford . Oxford University Press.

(3) SCOTT M. S. Introducción. En M. S. SCOTT (1990) *The Corporation of the 1990s* . New York Oxford . Oxford University Press.

(4) NAISBITT J., Megatendencias, Ed. 19

(5) VENKATRAMAN N. IT-Induced Business Reconfiguration . En M. S. SCOTT (1990) *The Corporation of the 1990s* . New York Oxford . Oxford University Press.

(6) ROTEMBERG J. J. y SALONER G., Interfirm competition and colaboration. En M. S. SCOTT (1990) *The Corporation of the 1990s* . New York Oxford . Oxford University Press.

(7) MACDONALD K. H. Business strategy development, Alignmetns, and Redesign . En M. S. SCOTT (1990) *The Corporation of the 1990s* . New York Oxford . Oxford University Press.

(8) BRADLEY S. P. , *The role of IT networking in sustaining competitive advantage.* En S. P. BRADLEY , J. A. HAUSMAN, R. NOLAN (1993), *Global Competition and technology.* Boston Massachusetts. Harvard Business School.

(9) IVES B. LEARMOUTH G., The information system as a competitive weapon (1987)

(10) OSTERMAN P. . The impact of IT on Jobs and Skills En M. S. SCOTT (1990) *The Corporation of the 1990s* . New York Oxford . Oxford University Press.

(11) Furth J., Carvell T., Managing in a wired company, Fortune, Julio 11 de 1994.

(12) ROCKART J. F. y SHORT J. E., The networked organization and the management ofInterdependence. En M. S. SCOTT (1990) *The Corporation of the 1990s* . New York Oxford . Oxford University Press.

(13) Harvard Business Review, The end of delegation ? Information technology and CEO, Harvard Business Review, Septiembre-Octubre 1995.

(14) MALONE T. W. ROCKART J. F. . ., *How* WillI information Technology reshape Organizations ? Computers as coordination techgnology. En S. P. BRADLEY , J. A. HAUSMAN, R. NOLAN (1993), *Global Competition and technology.* Boston Massachusetts. Harvard Business School.

(15) MALONE T. W. YATES J., BENJAMIN R., Electronic markets and electronic hierarchies, Communications ACM, Junio de 1987.

(16) ECCLES R. G. y NOLAN R.L., ., *A Framework for the Design of theEemerging GlobalOorganizational Structure* . En S. P. BRADLEY , J. A. HAUSMAN, R. NOLAN (1990), *Global Competition and technology.* Boston Massachusetts. Harvard Business School.

Un Lenguaje de Consultas sobre Estructura y Contenido en Bases de Datos de Texto

Gonzalo Navarro

Depto. de Ciencias de la Computación
Universidad de Chile
Blanco Encalada 2120 - Santiago - Chile
email: gnavarro@dcc.uchile.cl

Resumen

Las bases de datos textuales estructuradas son relativamente nuevas en el área de recuperación de información. No existe aun consenso acerca de cómo debería ser el modelo de estructuración. Las soluciones actuales se centran en los problemas o de expresividad o de eficiencia. Más aun, no hay un modelo formal y completo para analizar esta expresividad.

Nuestro objetivo es encontrar un modelo de estructuración y un lenguaje de consultas que sea suficientemente expresivo y eficientemente implementable. Para ello, se evaluaron las soluciones actuales, se definió un modelo con las características deseadas, se comparó su expresividad con los otros en términos teóricos y prácticos, se definieron algoritmos para implementarlo y se estudió su complejidad de peor caso en tiempo y espacio, y se desarrolló un prototipo para evaluar heurísticas y estudiar tiempos promedios en casos reales.

Este trabajo permite concluír que un lenguaje de consultas orientado a conjuntos y basado en operaciones sobre componentes estructurales cercanas de una o más jerarquías es suficientemente expresivo y eficientemente implementable. También presenta un marco para comparar expresividad, que da una idea de hasta dónde se puede enriquecer ésta sin degradar la performance. Por último, sugiere varias líneas de investigación futuras.

Abstract

Structured text databases are relatively new in the field of information retrieval. There is still no consensus on how the structuring model and the query language should be. Current approaches focus strongly either on expressivity or on efficiency issues. Moreover, there is no formal and complete foundation to analyze this expressivity.

Our goal is to find a structural model and a query language that is expressive and efficiently implementable. To achieve it, previous work was first evaluated, a model with the desired characteristics was defined, its expressivity was compared against the others, formally and practically, algorithms were defined and their worst-case space and time complexity analyzed, and a real prototype was developed, to evaluate heuristics and draw average running times.

This work leads to the conclusion that a set-oriented query language based on operations on nearby structure elements of one or more hierarchies is quite expressive and efficiently implementable. It also presents a framework to compare expressivity that gives an idea of up to where it can be enriched without degrading the performance. Finally, it suggests some research directions.

Este trabajo ha sido parcialmente financiado por los proyectos FONDECYT 1930765, 1940271 y 1950622.

1 Introducción

Las bases de datos textuales están recibiendo cada vez más atención, debido a sus múltiples aplicaciones: bibliotecas, automatización de oficinas, ingeniería de software, diccionarios automatizados, y en general cualquier problema basado en mantener y recuperar información textual [13, cap. 1].

El propósito de una base de datos textual es almacenar documentos de texto, estructurados o no. De modo que está compuesta de dos partes: contenido y estructura (si la hay). El contenido es el texto mismo, mientras que la estructura relaciona distintas partes de la base de datos con algún criterio.

El objetivo de cualquier sistema relacionado con recuperación de la información es ayudar a los usuarios de la base de datos a encontrar lo que buscan. Las bases de datos de texto no son como las relacionales [10], en las que la información está ya formateada y preparada para ser recuperada mediante "claves". La información está allí, pero no es fácil recuperarla. El usuario debe especificar lo que desea, ver los resultados, reformular la consulta, y así, hasta estar satisfecho con el resultado. Cualquier cosa que se haga para para ayudar a los usuarios a encontrar lo que buscan merece ser considerado.

Tradicionalmente, las bases de datos textuales han permitido a los usuarios buscar o en el contenido (palabras, frases, etc.) o en la estructura (por ejemplo, mirando el índice de un libro). Sin embargo, esto no es totalmente satisfactorio. Por ejemplo, una persona puede recordar que lo que quiere estaba escrito en itálicas, cerca de una figura que decía algo sobre "Chile". Buscar la palabra "Chile" puede ser una mala idea, como puede serlo buscar todos los textos en itálicas o todas las figuras del libro. Lo que realmente ayudaría sería un lenguaje que permitiera decir "quiero un texto en itálicas, cerca de una figura que contenga la palabra 'Chile' ". Esta consulta habla del contenido y de la estructura de la base de datos al mismo tiempo.

Mezclar el contenido y la estructura nos permite expresar consultas muy potentes, mucho más expresivas que las de cada mecanismo por separado. Usando un lenguaje que integre ambos tipos de consultas, podemos potenciar la calidad de recuperación de las bases de datos textuales. Sin embargo, ambos aspectos presentan sus propios desafíos.

Por un lado, la información almacenada como "contenido" no está formateada, sino en forma de lenguaje natural. Eso implica que ninguna metodología basada en información formateada (como el modelo relacional) es adecuada para extraer la información deseada; y que ninguna metodología que asuma objetos no interpretados y se base en sus atributos, también formateados (por ejemplo bases de datos multimedia [4]) es suficientemente potente para expresar el complejo modelo de información que el texto representa. La información debe ser extraída del texto, pero no en forma rígida (véase también [28]).

Por otro lado, no hay consenso acerca de cómo debería ser el modelo de estructuración de una base de datos. Hay varios posibles modelos, que van desde no estructurar en absoluto hasta complejas redes de relaciones. Decidir usar un modelo de estructuración implica elegir también qué clase de consultas se pueden hacer a la estructura. Hay un compromiso entre proveer un modelo de estructuración muy poderoso y poderlo implementar eficientemente.

El objetivo de este trabajo es presentar un modelo para estructurar y consultar bases de datos textuales, que sea suficientemente expresivo y eficientemente implementable. No existe en este momento, que sepamos, ninguna solución que satisfaga ambos objetivos. Para comparar expresividad, es necesario definir un marco en el que ubicar todos los modelos, que en este momento no existe. Para comparar eficiencia, es necesario describir los algoritmos que construyen los índices y procesan las consultas, analizar su complejidad en términos de espacio y tiempo, y obtener resultados experimentales a partir de un prototipo real.

La Tesis está organizada de la siguiente forma. En el Capítulo 2 se repasan los trabajos relacionados con el problema. En el Capítulo 3 se presenta informalmente nuestro modelo. En el Capítulo 4 se analiza su expresividad. En el Capítulo 5, se definen y analizan los algoritmos y estructuras de datos para implementar el modelo. En el Capítulo 6 se presenta una implementación real y se obtienen valores reales de tiempos promedios. Finalmente, en el Capítulo 7, se presentan las conclusiones y futuras líneas de investigación.

En este resumen se presenta una sección por cada capítulo del trabajo. Distintos aspectos de este trabajo has sido publicados en [27] (descripción del modelo), [3] (survey de modelos) y [26] (análisis de expresividad), y se han enviado para publicar una exposición de los algoritmos y una versión journal exponiendo el modelo.

2 Trabajos Relacionados

Cubriremos aquí los trabajos relacionados con el problema de consultar una base de datos textual. Veremos primero los modelos tradicionales y luego las ideas más novedosas. Una comparación más detallada se puede ver en [3].

2.1 Modelos Tradicionales

En esta sección se presentan cuatro líneas principales en la solución del problema, que constituyen los modelos "tradicionales" para encararlo. Estos modelos son: el modelo relacional [10] y sus extensiones a texto [33, 11], el modelo tradicional de recuperación de la información [31, 30, 32] (llamado "RI tradicional"), el modelo de texto completo [13, 2, 25], e hipertextos y lenguajes orientados a objetos [6, 17, 34, 7, 22].

Si bien estos modelos no son lo suficientemente potentes como para extraer la información que queremos de las bases de datos de texto, resuelven otros problemas que los lenguajes puramente orientados a la estructura no resuelven en general (por ejemplo, tuplas y joins, atributos, etc.).

En [28] se sugiere que es mejor poner una capa integrando una base de datos tradicional con una textual que pretender diseñar una base de datos que incluya todas las características. Por ejemplo, en [8] se muestra que las consultas relacionadas con la estructura se manejan mejor con una maquinaria que sepa acerca de la semántica de las jerarquías, que si se usa un lenguaje de base de datos orientado a objetos cualquiera.

Nosotros nos basamos en esta idea. Diseñamos un lenguaje orientado a explotar las características textuales y estructurales. Otras cosas, como tuplas y joins, deben añadirse mediante integrar este lenguaje con otro orientado a esa clase de operaciones, por ejemplo una base de datos relacional.

Por otro lado, no resolvemos el problema de unir consultas de estructura con otras que involucran operaciones como ordenamiento por relevancia (por ejemplo, obtener las secciones donde la palabra "computador" es relevante). Véanse en [28] algunas ideas sobre este tema.

2.2 Nuevos Modelos

Estos se caracterizan por dos hechos importantes: la estructura de los documentos es más o menos jerárquica (lo cual es razonablemente expresivo) y el lenguaje de consultas integra en una sintaxis uniforme las consultas sobre texto y sobre estructura. En el trabajo se cubren varios modelos, que aquí sólo mencionamos brevemente:

Modelo híbrido [2]: Combina campos (zonas del texto, como "título" o "apéndices") con un sistema tradicional de texto completo. Los campos se utilizan para restringir las búsquedas al texto dentro de algún campo dado. Toda respuesta a una consulta consiste de un conjunto de documentos y un conjunto de calces dentro de los documentos (donde se encontró lo que se pedía). Este es uno de los modelos más simples dentro de los que permiten mezclar consultas sobre contenido y estructura. Gracias a esa simplicidad, puede implementarse muy eficientemente.

PAT expressions [29]: En este modelo la estructura se define principalmente mediante expresiones de texto, de modo que en un texto marcado al estilo SGML se puede decir que una sección es lo que va desde donde hay un `<seccion>` hasta donde hay un `</seccion>` en el texto. Esto permite indizar en forma flexible, pero impone una estructura plana en los índices (una sección no podría contener otra sección). Gracias a estas restricciones, el modelo también puede implementarse muy eficientemente.

Listas solapadas [5]: Es una evolución a partir de PAT expressions, que resuelve los problemas semánticos de aquél (relacionados con la posibilidad de que surjan solapamientos de resultados en las operaciones) mediante permitir que los segmentos de una estructura solapen en el texto, aunque no que se contengan. Se obtiene mucha mayor expresividad y una implementación igualmente eficiente.

Listas de referencias [24]: Este modelo permite expresar, además de relaciones jerárquicas, conexiones estilo hipertexto entre los documentos y atributos, de modo que sólo se considera de él la parte relacionada con la estructura. La respuesta a una consulta es una lista de referencias (punteros) a la base de datos, de modo que se puede unir en forma elegante a listas de referencias obtenidas por mecanismos hipertextuales. Si bien permite una estructura jerárquica, las respuestas deben ser planas, y además del mismo tipo de constructor (por ejemplo, sólo secciones).

Parsed strings [15]: Este es en realidad un modelo de manipulación de datos, más que de consulta, de modo que se lo deja de lado al momento de comparar expresividad. Se basa en ver la base de datos jerárquica como descrita por una gramática libre de contexto (que representa el esquema de la base). Las operaciones realizan transformaciones sobre ése árbol, obteniéndose así un lenguaje muy flexible y potente, pero de implementación muy ineficiente.

Matching de árboles [21]: Este modelo consta de una única primitiva: la consulta es un árbol, y la respuesta a la consulta son todos los puntos de la estructura (árbol) de la base de datos donde la consulta calza, en el sentido de ser la raíz de un subárbol de la base donde la consulta puede ser embebida respetando los labels, las relaciones de descendencia y (opcionalmente) de orden entre hermanos. Esto se enriquece mediante un lenguaje estilo Prolog, con variables lógicas y búsqueda exhaustiva de todas las combinaciones de calces, para obtener un lenguaje de consulta potente, pero de difícil implementación [20]. Todo lo que involucre variables lógicas o matching sin respetar orden es NP-Completo, y fuera de eso, si bien es polinomial, debe recorrerse toda la base de datos buscando los calces.

3 Nuestro Modelo

3.1 Conceptos Fundamentales

En esta sección se exponen las ideas generales de cómo debería ser el modelo de estructuración y el lenguaje de consultas para alcanzar los objetivos de expresividad y eficiencia al mismo tiempo. Luego se define un modelo siguiendo estos principios. Una exposición completa del modelo se puede ver en [27].

Como queremos definir un lenguaje totalmente composicional, podemos considerar a las expresiones de consulta como árboles sintácticos, donde los nodos representan operaciones a realizar y sus subárboles los operandos.

Un primer punto es que queremos un lenguaje orientado a conjuntos, porque se han mostrado exitosos en otras áreas (como el modelo relacional), y porque si hay que extraer el conjunto completo de las respuestas, es posible encontrar algoritmos que recuperen los elementos a un costo muy bajo por elemento.

Para obtener el conjunto de respuestas queremos además evitar una solución "top-down", donde las respuestas se buscan en todo el árbol. Preferiremos una estrategia "bottom-up": la idea es poder encontrar rápidamente un conjunto pequeño de candidatos para las respuestas, y de ahí eliminar los que no cumplen con el criterio de búsqueda.

Nuestras operaciones toman y devuelven entonces conjuntos de nodos. Estos son subconjuntos del conjunto de todos los nodos del árbol de la base de datos. El único momento en que se resuelve una consulta de matching de texto o se designa por su nombre a un elemento de la estructura (por ejemplo, "sección") es en las hojas del árbol sintáctico de la consulta. Estas hojas deben ser resueltas con alguna clase de índice, y convertidas en un conjunto de nodos. De ahí en más, todos los operadores reciben y devuelven conjuntos de nodos (véase la Figura 1).

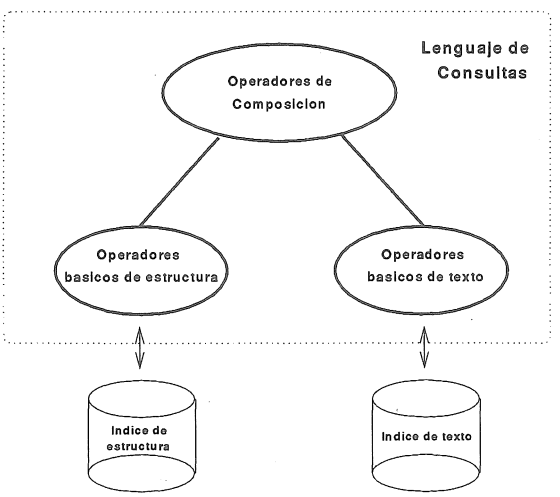


Figura 1: Diagrama inicial de cómo opera nuestro modelo.

Con ésta idea, usamos índices para recuperar los nodos que satisfacen una consulta de matching

de texto, o los nodos de una componente estructural dada (o "constructor"). Esos conjuntos deben ser obtenidos sin recorrer la base de datos completa.

Una vez que convertimos las hojas de la consulta en conjuntos, todas las otras operaciones toman esos conjuntos y los operan. Normalmente un conjunto va a contener los candidatos para el resultado de la operación. Obsérvese que de esta forma nunca debemos recorrer la estructura completa.

Por otro lado, estas operaciones de conjuntos pueden ser tan caras como recorrer la base de datos. Para evitar esto se imponen dos condiciones: que los nodos tengan un *segmento* (par de números representando una porción contigua del texto) asociado, y que las operaciones trabajen en función de los nodos con segmentos cercanos de ambos operandos.

La asociación de nodos con segmentos permite usar estructuras de datos eficientes para disponer los nodos basándose en los segmentos que representan (formando un árbol, por ejemplo). En otros modelos [21, 15], hay una conexión débil entre los nodos y el segmento que poseen en el texto, de modo que necesitan recorrer todo el árbol para encontrar lo que necesitan.

El operar con nodos cercanos permite aplicar técnicas de "dividir y reinar", para reducir drásticamente el área en la cual buscar los nodos a operar. De este modo se evitan las operaciones de "todos contra todos" como en [20], y obtener una implementación eficiente. Este esquema nos permite tener más de una estructura jerárquica, si son independientes.

Por otro lado, debemos mostrar que muchos operadores interesantes son del tipo que permitimos, en el sentido de operar con nodos cercanos y sólamente necesitar de la identidad de los nodos y del segmento asociado.

3.2 Modelo de Datos

Una base de datos textual se compone de dos partes:

- Texto, que se ve como una (larga) secuencia de símbolos. Que este texto se almacene como se lo ve, o que sea filtrado para ocultar marcado o componentes sin interés, no es importante para el modelo, dado que sólo usamos la visión lógica del texto. Además, los símbolos pueden ser caracteres, palabras, o lo que sea.

- Estructura, que se organiza como un conjunto de jerarquías independientes (ortogonales). Cada jerarquía tiene sus propios tipos de nodos, y las áreas cubiertas por los nodos de distintas jerarquías pueden solapar, si bien esto no puede ocurrir dentro de una misma jerarquía.

Nuestro modelo de utilización es que el texto es estático, las jerarquías se construyen sobre él una vez (o esporádicamente) y las consultas son frecuentes. Esto se tiene en cuenta a la hora de elegir qué operaciones implementar eficientemente.

Cada jerarquía (o árbol) se llama *view*, que como su nombre sugiere es una forma independiente de ver el texto (por ejemplo, capítulos/secciones/párrafos o páginas/líneas). Se entiende que la raíz de la view contiene toda la base de datos.

Cada view tiene una serie de *constructores*, que denotan tipos de nodos en el árbol correspondiente (por ejemplo páginas, capítulos, etc.). Los conjuntos de constructores de dos views distintas son disjuntos.

Cada nodo del árbol correspondiente a una view tiene un constructor asociado, y un *segmento*, que como se explicó antes es una porción contigua del texto. El segmento de un nodo debe incluír a los de sus hijos en el árbol (aunque no tiene por qué ser una inclusión estricta).

Cualquier conjunto de segmentos disjuntos se puede ver como perteneciente a una *view de texto* especial, en la que los nodos pertenecen a un *constructor de texto* y tienen una estructura plana (todos los nodos en el segundo nivel del árbol). Así, la *view de texto* tiene un nodo por cada posible segmento del texto. La idea es usar esa view para modelizar consultas de matching de texto, a las cuales imponemos que retornen una secuencia de segmentos disjuntos. Esta imposición no es esencial, dado que perfectamente podría generarse una jerarquía. Sin embargo, asumimos que la estructura es plana para algunas operaciones sobre consultas de texto, las que no serían aplicables en otro caso.

3.3 Lenguaje de Consultas

En esta sección definimos un lenguaje de consultas que opera sobre la estructura definida previamente, incluyendo también consulta por contenido. No pretendemos definir un lenguaje monolítico ni completo, dado que los requerimientos varían mucho según la aplicación. Incluír todas las alternativas en un mismo lenguaje de consulta lo haría demasiado complejo. En cambio, mostramos una cantidad de características que pueden ser útiles, de modo de seleccionar un subconjunto apropiado a cada necesidad.

Lo que es importante en este conjunto es que las operaciones sólo necesiten operar con nodos cercanos, y baciones en la identidad de los nodos y el segmento que les corresponde. Las operaciones que no cumplen con eso difícilmente se puedan implementar en forma eficiente.

Definimos un álgebra de operadores sobre conjuntos de nodos. Cada consulta retorna un conjunto de nodos del árbol de alguna jerarquía (no un subárbol), de modo que no se puede manipular subárboles ni ver las consultas como "views" (en el sentido relacional). En este sentido, el modelo es más débil que [15].

Decidimos no unir nodos de distintas views en un mismo resultado de consulta por dos razones: por un lado no queda claro, siendo las views formas independientes y distintas de ver el texto, si esto tiene algún sentido (por ejemplo, "capítulos o páginas con una figura"); y por otro, la implementación es mucho más eficiente si cada conjunto presenta una jerarquía estricta. En [5] se prefiere en cambio permitir solapamientos pero no anidamientos en las respuestas.

Si bien no es posible recuperar subárboles, el álgebra permite seleccionar nodos basándose en su *contexto* en la jerarquía, como [21]. Obsérvese que este lenguaje no es para usuarios finales, sino una representación operacional de las consultas.

En la Figura 2 se esquematizan las operaciones. Hay operaciones básicas de matching de texto y de obtención de constructores, operaciones para combinar los conjuntos resultantes de otras operaciones, para seleccionar elementos basándose en criterios de inclusión, cercanía y en relaciones de ascendencia y descendencia directa en la view. A continuación se presenta una descripción de las operaciones más comunes, en la Tesis se detallan varias más:

Sublenguaje de matching: Es el único que accede al contenido de la base de datos, y es ortogonal al resto del lenguaje.

Matches: El lenguaje de matches genera un conjunto de segmentos no solapados ni anidados, que se introducen en el modelo como pertenecientes a la *view de texto*, como se explicó. Por ejemplo `'computador'` genera el árbol plano de todos los segmentos donde esa palabra aparece en el texto. Obsérvese que el lenguaje de matches puede permitir expresiones mucho más complejas (cercanía, expresiones regulares, matching con errores, etc.).

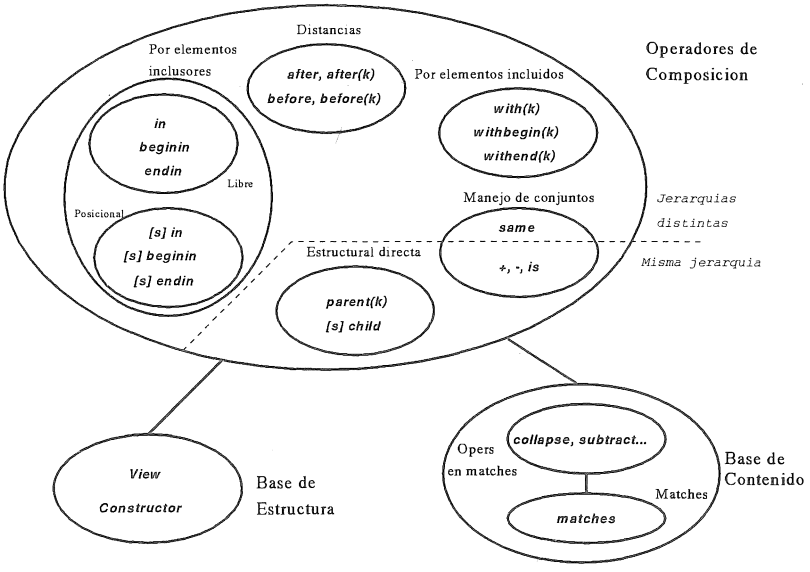


Figura 2: Las operaciones de nuestro modelo, clasificadas por tipo.

Operaciones sobre matches: Son aplicables sólamente a subconjuntos de la view de texto, y hacen transformaciones a los segmentos. Este punto y el anterior se consideran conjuntamente como el mecanismo para generar consultas de matching de texto, y no restringimos el lenguaje a ningún sublenguaje específico para esto. En la Tesis mostramos una cantidad de buenos candidatos para este conjunto, aquí incluímos sólo un ejemplo:

M collapse M': Es el conjunto de segmentos obtenidos al superponer ambos conjuntos de segmentos; si resulta un solapamiento se unen las dos áreas. Por ejemplo, `'Juan Perez' collapse 'J. Perez'` obtiene las apariciones de Juan Pérez en el texto, en sus dos formas.

Operadores básicos de estructura: Son la otra clase de hojas del árbol sintáctico de la consulta, que se refieren a componentes estructurales.

Nombre de constructor: (consultas "**Constr**"). Es el conjunto de todos los nodos de un constructor dado. Por ejemplo `capitulo` recupera todos los capítulos de un libro.

Nombre de view: (consultas "**View**"). Es el conjunto de todos los nodos de la view dada. Por ejemplo, `Formateado` recupera la view completa relacionada con aspectos de formateado. El mismo efecto se obtiene sumando (operador "+") todos los constructores de la view.

Operadores incluído-en: Seleccionan elementos del primer operando que están, en algún sentido, incluídos en el segundo.

Inclusión libre: Seleccionan cualquier elemento incluído, por ejemplo:

P in Q: Es el conjunto de todos los nodos de P incluídos en un nodo de Q. Por ejemplo, `citacion` **in** `tabla` recupera todas las citas hechas desde dentro de una tabla.

Inclusión posicional: Seleccionan sólo los elementos incluídos en una posición dada. Para definir posición, sólo se tiene en cuenta el top-level de los elementos incluídos en cada inclusor. Por ejemplo:

[s] P in Q: Lo mismo que **in**, pero sólo califican los nodos que descienden de un nodo de Q en la posición (de izquierda a derecha) considerada en s. Para linealizar la posición, para cada nodo de Q se considera sólo el top-level de los nodos de P no disjuntos con el nodo de Q, y los que solapan se descartan junto con sus descendientes. El lenguaje para expresar posiciones (valores de s) es también independiente. Consideramos que expresar uniones finitas de $i..j$, $último{-}i..último{-}j$, y $i..último{-}j$ alcanza para la mayoría de los propósitos. Por ejemplo, `[3..5]` `seccion` **in** `capitulo` recupera la 3ra, 4ta y 5ta sección de todos los capítulos. Si las secciones incluyeran otras secciones, sólo se considerarían las secciones top-level.

Operadores inclusores: Seleccionan los elementos del primer operando que incluyen en algún sentido elementos del segundo, por ejemplo.

P with(k) Q: Es el conjunto de nodos de P que incluye al menos k nodos de Q. Si (k) no está presente, se asume 1. Por ejemplo, `seccion` **with**(5) `'computador'` selecciona las secciones en las que la palabra "computador" aparece 5 veces o más.

Operadores de estructura directa: Seleccionan elementos del primer operando basados en criterios estructurales directos, es decir por relaciones de padre/hijo directos en el árbol de la view.

[s] P child Q: Es el conjunto de nodos de P que son hijos (en el árbol de la view) se algún nodo de Q, en una posición considerada en s (o sea, s-ésimos hijos). Ambos conjuntos deben ser de la misma view, que no puede ser la view de texto. Si $[s]$ no está presente, se asume $1..último$. Por ejemplo, `titulo` **child** `capitulo` recupera los títulos de todos los capítulos (y no títulos de secciones dentro de los capítulos).

P parent(k) Q: Es el conjunto de nodos de P que son padres (en el árbol de la view) de al menos k nodos de Q. Ambos conjuntos deben ser de la misma view, que no puede ser la view de texto. Si (k) no está presente, se asume 1. Por ejemplo, `capitulo` **parent**(3) `seccion` selecciona capítulos con 3 o más secciones.

Operadores posicionales: Seleccionan del primer operando los elementos que están a una cierta distancia de algún elemento del segundo operando, bajo ciertas condiciones adicionales. Por ejemplo:

P after/before Q (C): Es el conjunto de nodos de P cuyos segmentos empiezan/terminan después/antes que el fin/comienzo de un segmento de Q. Si hay más de un candidato de P para un mismo nodo de Q, se considera el más cercano (si están a la misma distancia, entonces uno de ellos incluye al otro y se elige el más alto). Para que un nodo de Q se considere candidato para un nodo de P, el nodo minimal de C que lo contiene debe ser el mismo que el del nodo de P, o no existir en ambos casos. Por ejemplo, `tabla` **after** `figura` (`capitulo`) recupera las tablas más cercanas a una figura que las precede, dentro del mismo capítulo.

Operadores de manipulación de conjuntos: Manipulan ambos operandos como conjuntos, implementando unión, diferencia, intersección, etc. Ejemplos:

P – Q: Es el conjunto diferencia entre P y Q. Ambos deben ser de la misma view, y esta view no debe ser la de texto. Por ejemplo, `capitulo – (capitulo with figura)` son los capítulos sin figuras.

P same Q: Es el conjunto de nodos de P cuyo segmento es el mismo de algún nodo de Q. P y Q pueden ser de diferentes views. Por ejemplo, `titulo same 'Introduccion'` recupera los títulos que dicen (exactamente) "Introducción".

En la Figura 3 se puede ver un ejemplo de una consulta.

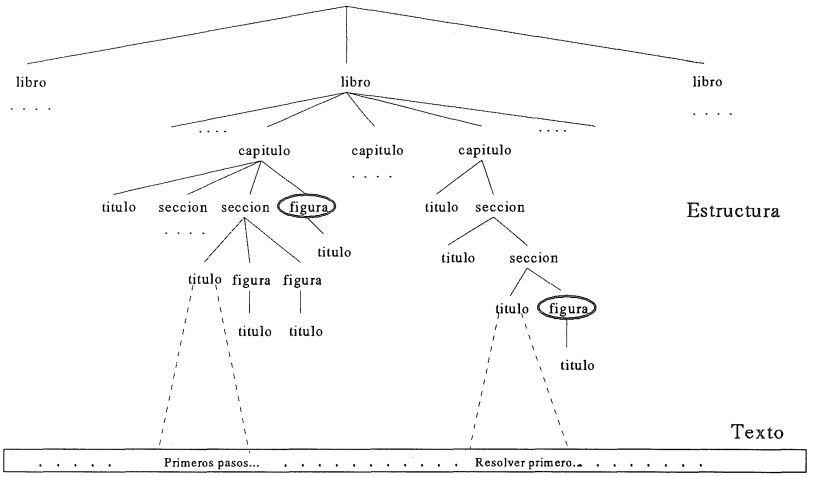


Figura 3: Efecto de la consulta `[ultimo] figura in (capitulo with (seccion with (titulo with 'primer*')))`. Los círculos destacados son los nodos seleccionados.

3.4 Una Arquitectura de Software

En esta sección se describe una posible arquitectura de software para un sistema basado en nuestro modelo.

Los usuarios interactúan con el sistema vía una interfaz, en la cual definen lo que quieren en algún lenguaje amigable. Esa interfaz debe entonces convertir esa consulta en nuestro árbol sintáctico, es decir en el lenguaje que se ha definido. El árbol es entonces enviado al módulo de consulta.

Este módulo optimiza la consulta y genera un plan de acción inteligente para evaluarla (linealiza el árbol en una secuencia de operaciones a realizar). Las hojas del árbol de la consulta involucran extraer componentes de la jerarquía por su nombre (constructores) y expresiones de matching de texto. Las primeras se resuelven accediendo al índice de estructura para obtener el conjunto completo de nodos de ese constructor (un conjunto de identificadores de nodos y sus segmentos).

Las segundas se envían al módulo de texto, que retorna una lista de segmentos correspondientes a las porciones del texto que calzaron. De ahí en adelante, el resto de las operaciones se efectúan internamente, hasta que se entrega el resultado final (un conjunto de nodos) a la interfaz.

La interfaz está a cargo de la visualización los resultados. Para ello debe acceder al contenido de la base de datos, a las porciones del texto indicadas por los segmentos a mostrar. Esto se hace a través de un pedido al módulo de texto, dado que sólo él sabe cómo acceder al texto. Este módulo es el único responsable de de la visión que tenemos sobre el texto, y el que puede filtrar marcas y stopwords, reemplazar sinónimos, etc.

Véase en la Figura 4 un diagrama de cómo sería un sistema basado en este esquema. La "capa documental" está pensada para soportar un manejo de documentos más sofisticado, como colecciones de documentos. Traslada todas las referencias a colecciones (como las vería el usuario) a referencias a documentos (como están indizados en niveles inferiores).

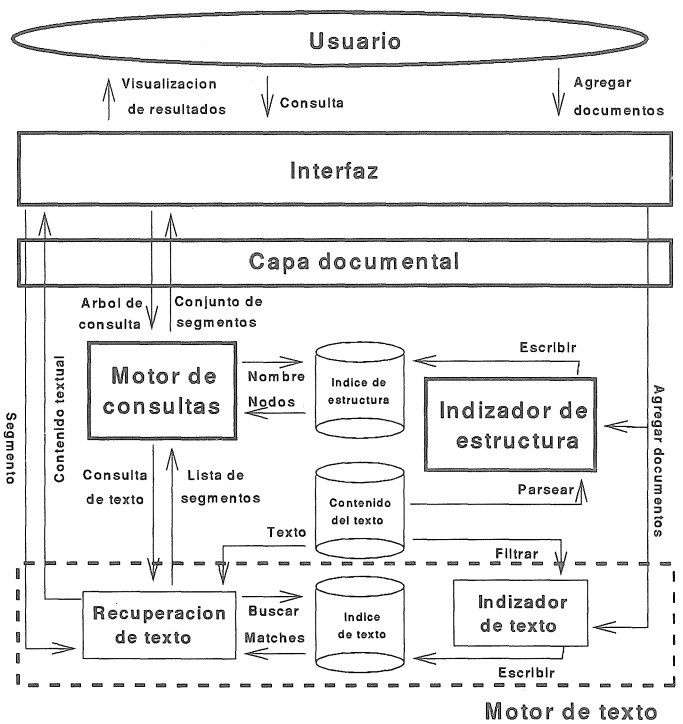


Figura 4: La arquitectura de un sistema según nuestro modelo. Las flechas muestran el proceso de resolver una consulta (lado izquierdo) o de incorporar documentos (lado derecho).

La Figura también muestra el proceso de indización. Si el módulo de texto está completamente separado, puede haber dos procesos de indización. Uno de ellos (realizado en el módulo de texto) indiza el texto para responder a preguntas de matching de texto. El otro extrae la estructura de alguna forma a partir del texto (parsing, reconocimiento de marcas, etc.), y crea el índice de estructura, que es accedido más tarde por el módulo de consultas. De hecho, ambos indizadores

deben colaborar, pues el marcado usado por el segundo para reconocer la estructura (si lo hay) debe ser filtrado por el primero cuando se presenta el texto a las capas superiores.

4 Expresividad

En este capítulo de la Tesis se estudia la expresividad que tiene nuestro modelo en relación con otros similares. Un análisis más detallado de la expresividad se puede ver en [26].

Lo primero que se hace es definir formalmente nuestro modelo de datos y las operaciones, de modo de tener una definición precisa para poder compararlo con otros.

Luego se procede a comparar formalmente el modelo con cada uno de los estudiados antes, salvo *p-strings*, que es un modelo de manipulación más que de consulta. Esta comparación procede mediante definir mapeos que lleven de expresiones de un lenguaje a expresiones del otro. De esta forma se puede tener una idea de cuáles son las cosas de un lenguaje que pueden ser representadas en el otro, y cuáles no. Otro parámetro interesante es cuán largas pueden resultar las consultas traducidas con respecto a las originales, y si se necesita más información en los índices.

De esta comparación de nuestro modelo con cada uno de los otros, se concluye que nuestro modelo es superior en la mayoría de los aspectos al modelo híbrido [2], a PAT expressions [29], a listas solapadas [5] y a listas de referencias [24], mientras que matching de árboles [21] es superior en general al nuestro.

De esta comparación surgen ciertos criterios acerca de cómo debería ser analizado un lenguaje de este tipo. Se define una metodología para analizar estos lenguajes, que se subdivide en tres aspectos: mecanismo de estructuración, consulta por contenidos y consulta por estructura. La metodología se ve como una serie de "preguntas" que hay que hacerse a la hora de medir la expresividad de un lenguaje de este tipo.

A partir de considerar los modelos similares con esta metodología se obtiene un diagrama donde se muestran los temas más importantes a considerar como puntos, y los modelos como conjuntos que contienen los puntos si es que los incluyen adecuadamente (obsérvese que se trata de consideraciones, hasta cierto punto, informales). Con ésto se puede apreciar cuánto de lo importante incluye cada modelo (Figura 5).

5 Implementación

En esta sección se describen brevemente las ideas expuestas en el trabajo para implementar el modelo, y la eficiencia resultante.

Dado que el lenguaje es composicional, el problema es, dados los conjuntos de nodos resultantes de las sub-operaciones, cómo obtener el conjunto resultante de la operación a aplicar.

Una primera y fundamental disyuntiva es si usar evaluación completa ("full") o retardada ("lazy"). Evaluación full significa, antes de aplicar el operador actual, obtener completos los conjuntos sobre los que operar, y luego generar completo el conjunto resultante. Evaluación lazy significa tratar de obtener el resultado del operador más alto elemento por elemento, donde la extracción de cada elemento genera la necesidad de obtener elementos de los operandos, y así hasta las hojas, que van evaluando lo que se les va pidiendo. De esta forma, sólo la parte necesaria de los conjuntos resulta evaluada.

Considérese, por ejemplo, la consulta "`seccion with(10) 'Computador' `". Si, en vez de obtener toda la jerarquía de secciones y luego descartar las que no contienen 10 veces la palabra "Computador", se obtienen primero las secciones top-level, se descartan las que no contienen la

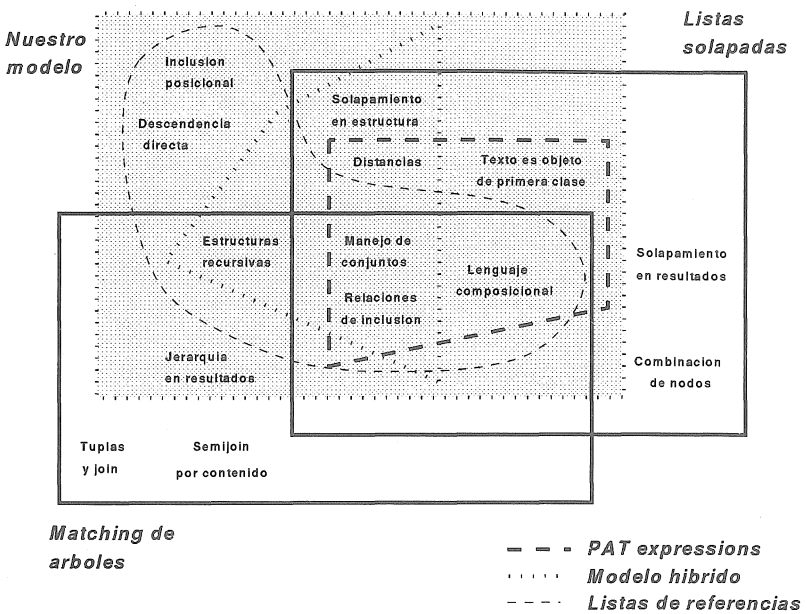


Figura 5: Representación gráfica de la comparación entre modelos.

palabra 10 veces y sólo se expanden las que sí la contienen, nos ahorramos de expandir inútilmente las secciones que resultan eliminadas en el primer paso.

Por otro lado, la evaluación lazy puede forzarnos a usar algoritmos menos eficientes, de modo que no es inmediato cuál alternativa es mejor. Es se determina experimentalmente.

Para evaluación full, se proponen dos técnicas distintas, que llamamos *search-operate* (buscar y operar) y *merge* (mezclar). La última se modifica para obtener evaluación lazy.

Finalmente, se pueden usar distintas clases de índices, algunos soportando mejor ciertas operaciones que otros, al costo de cierto requerimiento de espacio.

5.1 Indización

En esta sección de la Tesis se estudian distintas alternativas para implementar los índices. Todos los índices representan, de alguna forma, al árbol de la view. Dado que la view es estática, este árbol se mapea sobre un arreglo, nivel por nivel, de modo de ahorrar espacio de punteros y de algunas medidas que resultan implícitas si se pone nivel por nivel. La estructura de datos, cualquiera que sea, debe permitir hallar eficientemente el hermano y el primer hijo de cualquier nodo.

Primeramente se considera un índice completo, es decir, que permita realizar eficientemente todas las operaciones descritas en el modelo. Este índice ocupa 7 palabras por nodo, por lo cual deben estudiarse formas de mantenerlo en el disco. El problema con el disco es que si se almacena el arreglo de la view completo puede realizarse mucho seek para recuperar los nodos de un constructor. Para solucionar esto se propone tener un puntero al próximo elemento en el disco del mismo constructor (reemplazando otro puntero que se revela prescindible en la práctica),

o tener un archivo separado por constructor. Esta última técnica es mejor para consultas tipo **Constr**, pero peor para las tipo **View**. Se considera que las primeras son mucho más comunes. Finalmente, algunas consideraciones de reindización indican que sería beneficioso tener un archivo separado para cada nivel del árbol (en cualquier esquema).

Otro caso en que se puede acceder en forma aleatoria al índice en disco es para consultar si un nodo dado es padre directo de otro (para operaciones **parent** y **child**). Estos accesos aleatorios se pueden evitar mediante determinar sintácticamente (en la consulta) qué nodos pueden llegar a necesitar información de padre/hijo, y leer esa información al momento de resolver las hojas, cuando se leen esos nodos del disco y almacenar esa información no cuesta nada.

Si se tiene que no puede haber dos segmentos iguales en la jerarquía y se descartan las operaciones **parent** y **child**, es posible obtener un índice de sólo dos palabras por nodo, almacenando un arreglo con las posiciones iniciales de los segmentos y otro con las finales, ordenadas. Este índice no es bueno para evaluación lazy, porque obliga a expandir todo un subárbol para obtener sólo los hijos de un nodo (la operación normal de evaluación lazy). Para este caso es preferible eliminar información en el esquema de índice completo con un arreglo por constructor, para obtener un índice de tres palabras por nodo.

Para tener una idea de los tamaños involucrados, si se indiza código C con todos los detalles finos de parsing, un archivo de 1 Mb de código daría lugar a una estructura de 300 mil nodos, o sea 8 Mb para el índice completo y 2.3 Mb para el parcial. En cambio, si se indiza la estructura de un libro normal hasta los párrafos, un libro de 300 páginas (aproximadamente 1 Mb de texto) puede dar lugar a unos 1000 nodos, necesitando 28 Kb el índice completo y 8 Kb el parcial.

5.2 Consultas

En esta sección de la Tesis se detallan los algoritmos para efectuar cada operación y su análisis de tiempo en el peor caso. Primeramente se describe la estructura de datos con la que se almacenan los conjuntos de nodos. Esta estructura resulta ser un árbol también, en que las relaciones de descendencia se definen justamente teniendo en cuenta las posiciones en la view. Es decir, un nodo desciende de otro en el árbol si y sólo si desciende en la view (véase Figura 6).

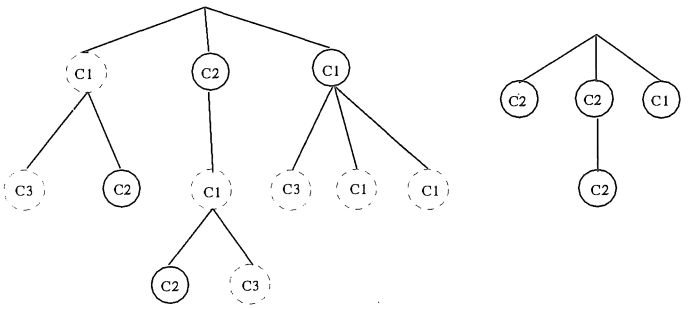


Figura 6: Ejemplo de una consulta (subconjunto de la view) y su representación como árbol.

Seguidamente se estudia el problema de generar un plan de acción inteligente a partir del árbol de la consulta, atendiendo a problemas como minimizar el espacio máximo que se requiere para mantener los resultados intermedios, políticas de swapping cuando no todos los resultados

Operación	Search-Operate	Merge	Lazy
$+/-$	$n \log d..n \log^2 d$	n	$n \min(d, h)$
same	$n \min(n, h \log d)$	n	n
in	$n \log d$	$\min(n, d^2 h)$	$\min(n, d^2 h)$
$[s]$**in**	$n \min(n, d + h \log d)$	$n \min(d, h)$	$n \min(d, h)$
with(k)	$n \min(n, h \log d)$	$k = 1 \ ? \ n \ : \ nh$	$n \min(n, k + dh)$
$[s]$**child**	$n \log d..n \log^2 d$	n	n
parent(k)	$n \log d..n \log^2 d$	n	$n \min(d, h)$
after/before(C)	$n \min(n \log d, dh)$	$n \min(n, dh)$	$n \min(n, dh)$

Tabla 1: Complejidad para las distintas versiones de algunos de los operadores estudiados en la Tesis.

intermedios caben en memoria, compartir conjuntos obtenidos por subexpresiones comunes, etc. Este tema deja en general una serie de líneas futuras de investigación que no se resuelven en este trabajo.

No vamos a reproducir aquí la descripción de los algoritmos. Sólo vamos a explicar la filosofía general de las operaciones tipo *search-operate*, *merge* y *lazy*.

En la primera, se selecciona el menor de los dos operandos, y para cada elemento E de su top-level, se busca este elemento dentro del segundo (los niveles se mantienen en arreglos para permitir búsqueda binaria en cada nivel), hasta hallar la zona donde debe operar. Además de operar en el lugar, el subárbol del elemento E debe a su vez ser operado con el subárbol de la zona encontrada por E. Puede darse que en los subproblemas haya cambiado la relación de tamaño y se elija buscar el segundo operando en el primero.

En las versiones *merge*, se recorren ambos árboles en forma sincronizada, buscando en cada caso contra qué elementos debe ir operando el elemento corriente de uno de los operandos. El método se llama así por la semejanza con el merging de dos listas ordenadas, donde se las recorre en forma sincronizada.

Las versiones *lazy* son muy similares a las *merge*, sólo que se expande un nivel del resultado por vez, de modo que a veces en vez de realizar todo el trabajo, se deja indicado para ser resuelto más adelante, si es requerido. Esto hace que haya que re-atravesar algunas listas, y la complejidad sea más alta que para las versiones *merge* correspondientes.

Esto es sólo una descripción general, los algoritmos son variaciones sobre la idea inicial. La descripción de los algoritmos se encuentra a todo lo largo del Capítulo 5. En cada caso se analiza el costo del algoritmo en el peor caso, basándose en tres parámetros de los operandos: n (el tamaño del árbol), h (la altura máxima) y d (la aridad máxima). En el caso de *lazy*, también se analiza cuánto se puede llegar a trabajar en una sóla invocación lazy, y cuánto se puede llegar a expandir de ambos operadores.

Véase en la Tabla 1 los resultados resumidos de las complejidades de todos los algoritmos. Como puede verse, las versiones *merge* son casi invariablemente las mejores, siendo lineales en la mayor parte de los casos.

6 Un Prototipo

Esta sección describe brevemente el prototipo desarrollado para probar el modelo en situaciones reales. El prototipo sigue el modelo propuesto para la arquitectura.

El motor de texto se basa en la API de SearchCity [1], un sistema de búsqueda en texto que implementa parcialmente el modelo híbrido [2] usando PAT o suffix arrays [13, cap. 5] y [25].

Para indizar la estructura, se crearon indizadores ad-hoc para varios "lenguajes" de estructuración: SGML [18, 14], LaTeX [23], código C [19] y DDIF [12]. En cada caso debió implementarse un parser que obtuviera la estructura a partir del texto y un filtro que eliminara las marcas de estructura al verse el texto desde un nivel más alto. Se implementó la versión más simple del índice de estructura (completo y en memoria).

El motor de consultas incluye todos los operadores representativos, en sus tres versiones (*search-operate*, *merge* y *lazy*).

Finalmente, la interfaz y la capa documental son aun rudimentarias.

Con este prototipo se condujeron varias pruebas, las más importantes son:

- Una prueba operador por operador, con tamaños de operandos entre 100 y 10.000 nodos, con operadores de árboles planos y profundos. Esto totalizó un total de 1300 pruebas. El objetivo principal era conocer los tiempos promedio de las operaciones y cuándo es mejor utilizar *merge*, *search-operate* o evaluación *lazy*.

- Una prueba de *lazy* vs *merge*, con diez consultas reales, más complejas que las de un sólo operador hechas en la prueba anterior. Se pretendía determinar cuánto tarda la evaluación *lazy* con respecto a *merge* en casos reales, y qué porcentaje del total de nodos eran expandidos en promedio.

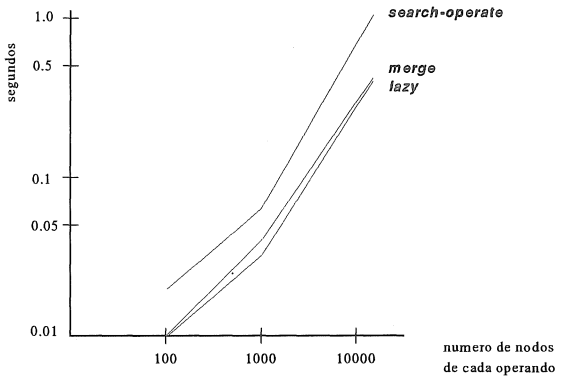


Figura 7: Tiempos típicos para operandos de igual tamaño. Observar que se usa una escala logarítmica.

La Figura 7 muestra los tiempos para una caso típico con un operador. Los tiempos corresponden a una Sun SparcClassic con 16 Mb de RAM, corriendo SunOS 4.1.3_U1. La velocidad de CPU es aproximadamente 26 SpecMark. Los principales resultados que se extraen de las pruebas son los siguientes:

- Los algoritmos *merge* son mucho mejores que los *search-operate*, para operandos de más de 1000 nodos (con menos nodos son similares). Los algoritmos *search-operate* han resultado de peor complejidad, más complejos de escribir y de analizar, y finalmente de peor comportamiento real: mayor promedio y varianza. De modo que deben ser definitivamente descartados.

- Los algoritmos *merge* y *lazy* son decididamente lineales en el tamaño de los operandos, y los *merge* sumamente predecibles en sus tiempos. Esto es bueno para tareas interactivas. Las versiones *lazy*, en cambio, tienen una varianza mucho mayor, aunque en promedio están cercanas a las *merge*.

- Los tiempos de *merge* para operaciones simples corresponden a unos 50.000 nodos procesados por segundo.

- En consultas más complejas (y más reales) las versiones *lazy* son generalmente mejores que las *merge*, aunque la fracción es sumamente variable. Un intervalo de confianza razonable es [0.25..0.90] (fracción de lo que tarda *lazy* con respecto a *merge*), y un valor común el 0.5. El porcentaje de nodos expandidos es también muy variable, yendo del 40% al 100% del total.

- En consultas complejas, el tiempo del *merge* es proporcional a la suma de los tamaños de todos los resultados intermedios. Una estimación más gruesa es $(2q+1) \times$ *tamaño promedio de operando*, donde q es la cantidad de nodos del árbol de la consulta. La constante en nuestra máquina resulta ser de $[4..7] \times 10^4$ nodos procesados por segundo.

7 Conclusiones y Trabajo Futuro

Se ha analizado el problema de consultar una base de datos de texto por su contenido y su estructura. Los modelos existentes se encontraron demasiado enfocados en expresividad o en eficiencia, pero no en ambas.

Se definió un modelo para estructurar documentos y un lenguaje de consulta que resultaron suficientemente expresivos y eficientemente implementables. La idea principal es una organización jerárquica de los nodos y un lenguaje de conjuntos basado en operaciones sobre nodos cercanos.

Para evaluar la expresividad se comparó el modelo propuesto con otros afines, mostrando qué aspectos de cada uno podía ser representado con el otro; y se definió una metodología para evaluar expresividad en modelos afines.

Para evaluar eficiencia se analizó el comportamiento de nuestros algoritmos en términos de tiempo y espacio, y se implementó un prototipo para obtener tiempos promedios y evaluar las distintas soluciones propuestas.

Los resultados de este trabajo son un nuevo modelo para estructurar y consultar bases de datos de texto, que tiene una implementación eficiente y buena expresividad, una metodología para comparar expresividad, y un estudio teórico y práctico del comportamiento del modelo en términos de tiempo y espacio.

Si bien se pueden agregar muchas operaciones al lenguaje, este trabajo muestra que algunas no pueden agregarse sin degradar la performance (por ejemplo, semijoin por contenidos). Finalmente, se insiste en que el lenguaje es un álgebra operacional y no está diseñado para usuarios finales. Sirve como representación intermedia de las operaciones.

Véase en la Figura 8 una comparación gráfica (e informal) de la ubicación de los distintos modelos, atendiendo a la eficiencia y a la expresividad.

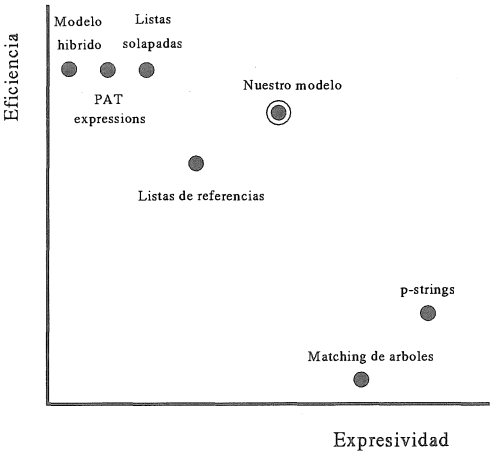


Figura 8: Comparación entre modelos similares, atendiendo a la eficiencia y la expresividad.

Este trabajo da un paso en la dirección de obtener una perspectiva unificadora acerca de cómo debería ser un lenguaje de consultas para bases de datos textuales, qué poder expresivo debería tener y cuán bien se puede implementar. También deja abiertas varias líneas de investigación que merecen ser seguidas:

- Encontrar más operaciones eficientemente implementables en nuestro modelo.
- Definir un lenguaje de consultas orientado a usuarios finales, para mapear sobre nuestra álgebra operacional.
- Refinar el mecanismo lazy, permitiendo expandir sólo partes de un nivel, dado que en este momento el mecanismo expande de una vez todos los hijos de un nodo y eso no es bueno para árboles poco profundos (y la mayoría lo son).
- Implementar las técnicas propuestas para mantener el índice y los operandos en disco, para estudiar el comportamiento de diferentes métodos y políticas.
- Diseñar un parser de consultas que genere planes de acción inteligentes, implementando las técnicas propuestas (árboles, DAGs y políticas de dealocación).
- Extender el prototipo, convirtiéndolo en una aplicación real, con manejo de índices en disco, swapping, generación de planes de acción, etc. Una posibilidad interesante es usar el mecanismo lazy como el motor de un sistema que combine consultas de conjuntos con navegación.

Para terminar, indicamos una serie de problemas abiertos que han aparecido a lo largo de este trabajo, y merecen un serio estudio aparte. Algunas son:

- El diseño de técnicas de optimización de consultas para nuestro lenguaje, basado en identidades algebraicas o en atomizar operaciones compuestas muy comunes.

- La integración entre este tipo de modelo y otros, como el relacional o el de RI tradicional. Este tema no se ha considerado aquí, dado que nos centramos en el problema de la estructura. Relacionado con esto está generalizar el problema para manejar estructuras no jerárquicas, como un hipertexto. No se ha hallado aun un lenguaje expresivo y eficiente para consultar estas estructuras.

- Hallar un modelo formal en que comparar expresividad. En este trabajo se hizo algo en ese sentido, pero el objetivo a largo plazo es una jerarquía formal y bien fundada. Véase [9, 16] como ejemplos. Otro problema es obtener consenso sobre un estándar práctico de expresividad, lo que es un problema de maduración.

Referencias

[1] Ars Innovandi, Santiago, Chile. *Search City 1.1. Text Retrieval for Windows Power Users*, 1992.

[2] R. Baeza-Yates. An hybrid query model for full text retrieval systems. Technical Report DCC-1994-2, Dept. of Computer Science, Univ. of Chile, 1994. `ftp://sunsite.dcc.uchile.cl/-pub/users/rbaeza/hybridmodel.ps.gz`.

[3] R. Baeza-Yates and G. Navarro. *Integrating contents and structure in text retrieval. ACM SIGMOD Record*, Mar. 1996. To appear. `ftp://sunsite.dcc.uchile.cl/pub/users/-gnavarro/sigmod.ps.gz`.

[4] E. Bertino, F. Rabitti, and S. Gibbs. Query processing in a multimedia document system. *ACM TOIS*, 6(1):1–41, Jan. 1988.

[5] C. Clarke, G. Cormack, and F. Burkowski. An algebra for structured text search and a framework for its implementation. *The Computer Journal*, 1995.

[6] J. Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41, Sept. 1987.

[7] M. Consens and A. Mendelzon. Hy$^+$: A hygraph-based query and visualization system. In *Proc. ACM SIGMOD'93*, pages 511–516, 1993. Video presentation summary.

[8] M. Consens and T. Milo. Optimizing queries on files. In *Proc. ACM SIGMOD'94*, pages 301–312, 1994.

[9] M. Consens and T. Milo. Algebras for querying text regions. In *Proc. PODS'95*, 1995.

[10] C. Date. *An Introduction to Database Systems.* Addison-Wesley, 6th edition, 1995.

[11] B. Desai, P. Goyal, and S. Sadri. A data model for use with formatted and textual data. *Journal of ASIS*, 37(3):158–165, 1986.

[12] Digital Equipment Corporation. *CDA - DDIF Technical Specification*, June 1991.

[13] W. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, 1992.

[14] C. Goldfarb. *The SGML Handbook*, 1990.

[15] G. Gonnet and F. Tompa. Mind Your Grammar: a new approach to modelling text. In *Proc. VLDB'87*, pages 339–346, 1987.

[16] M. Gyssens, J. Paredaens, and D. V. Gucht. A grammar-based approach towards unifying hierarchical data models. In *Proc. ACM SIGMOD'89*, pages 263–272, 1989.

[17] R. Hull and R. King. Semantic database modelling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, 1987.

[18] International Standards Organization. *Information Processing — Text and Office Systems — Standard Generalized Markup Language (SGML)*, 1986. ISO 8879-1986.

[19] B. Kernighan and D. Ritchie. *The C Programming Language.* Prentice-Hall, 1978.

[20] P. Kilpeläinen and H. Mannila. Grammatical tree matching. In *Proc. CPM'92*, pages 162–174, 1992.

[21] P. Kilpeläinen and H. Mannila. Retrieval from hierarchical texts by partial patterns. In *Proc. ACM SIGIR'93*, pages 214–222, 1993.

[22] W. Kim and F. Lochovski, editors. *Object-Oriented Concepts, Databases and Applications.* Addison-Wesley, 1989.

[23] L. Lamport. LaTeX: *A Document Preparation System.* Addison-Wesley, 1986.

[24] I. MacLeod. A query language for retrieving information from hierarchic text structures. *The Computer Journal*, 34(3):254–264, 1991.

[25] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. In *Proc. ACM-SIAM'90*, pages 319–327, 1990.

[26] G. Navarro and R. Baeza-Yates. Expressive power of a new model for structured text databases. In *Proc. PANEL'95*, pages 1151–1162, 1995. `ftp://sunsite.dcc.uchile.cl/pub/-users/gnavarro/clei.ps.gz`.

[27] G. Navarro and R. Baeza-Yates. A language for queries on structure and contents of textual databases. In *Proc. ACM SIGIR'95*, pages 93–101, 1995. `ftp://sunsite.dcc.uchile.cl/-pub/users/gnavarro/sigir.ps.gz`.

[28] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel. Database systems for structured documents. In *Proc. ADTI'94*, pages 272–283, 1994.

[29] A. Salminen and F. Tompa. PAT expressions: an algebra for text search. In *COMPLEX'92*, pages 309–332, 1992.

[30] G. Salton. Another look at automatic text retrieval systems. *CACM*, 29:648–656, 1986.

[31] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[32] G. Salton and M. McGill. *Automatic text processing.* Addison-Wesley, 1989.

[33] M. Stonebraker, H. Stettner, N. Lynn, J. Kalash, and A. Guttman. Document processing in a relational database system. *ACM TOIS*, 1(2):143–158, Apr. 1983.

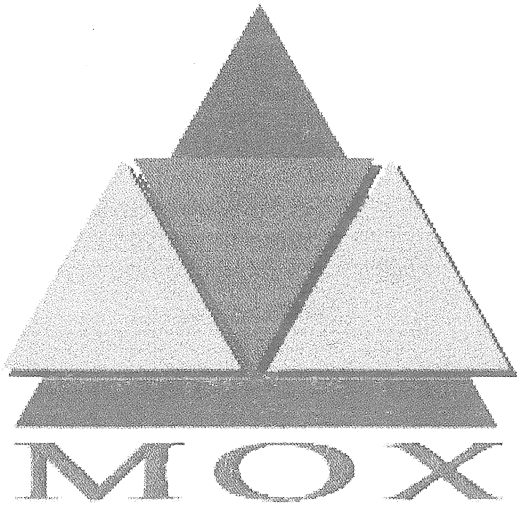
[34] J. Tague, A. Salminen, and C. McClellan. Complete formal model for information retrieval systems. In *Proc. ACM SIGIR'91*, pages 14–20, 1991.

Índice por autor

Abásolo, José 442
Abreu, Mario 647
Aguilar, José 226
Almansa, Andrés 659
Alvarado, Marco 28
Alvarez, Hernán 1021
Ambrosio, Ana Paula..... 430
Anciano, Juan 214
Antoneti, Jenny 845
Aráoz, Julián..... 359
Araujo, Flavio 833
Arias, Tatiana 967
Arráiz, Emely 214, 371
Ayala, Mauricio 322
Baeza-Yates, Ricardo 335, 671
Barán, Benjamín 715
Barra, Carlos 702
Barráez, Daniel 203
Barrero, Daniel 689
Barros, Ricardo 139
Baum, Gabriel 310, 857
Becker, Karin 1
Belchior, Arnaldo 63
Bello, Humberto 553
Benítez, Diana 715
Bergengruen, Olaf 659
Bezerra, Hednilson 152
Bezerra, Luiz 417
Bodart, François 1
Brito, Ma. Rosa 347
Camargo, Francisco 482
Campos, Gilda 647
Cañas, Javier 589
Castilho, José 468
Castro, Harold 405
Castro, Orlando 1057
Cayssials, Ricardo 1092, 1101
Chen, Fan..... 613
Cobo, Hernán 979
Concha, Mauricio 953
Cortés, Mauricio 87
Corzo, Fabio 238
Curbelo, Rosario 659
Curiel, Mariela 371
da Costa, Celso 739
da Fonseca, Fernando 152, 625
da Rocha, Ana 63, 647
da Rocha, Antônio 276
da Silva, Fabio 991
de Moraes, Rodrigo..... 75
Delahaye, Jean 250
Delorme, Charles 347
Di Serio, Angela 262
Díaz, Javier 637
Díaz, Alicia 882
Dietrich, Elton 494
Diverio, Tiarajú 1004
Domínguez, Ramón 203
dos Santos, Simone 991
Drets, Gustavo 659
Echagüe, Juan 727
Feldens, Miguel 468
Fematt, Rafael..... 751
Fernández, María C. 177
Ferro, Edgardo 1092, 1101
Fialco, Gabriel 727
Franco, Guillermo 751
Franco, José 922
Franky, Ma.Consuelo 1057
García, Eduardo 1045
Georges, Fouad 430
Germinal, Isern 808
Godoy, Ma. Alejandra 760
Gómez, Rafael 238
González, Marco 14
Gordillo, Silvia 882
Guimarães, Ricardo 139
Guzmán, Juan C. 517
Harari, Ivana 637
Hernández, Arturo 99
Hernández, Marcela 115
Hernández,Domingo 454
Hernández, José T. 689, 772
Herrero, Jorge 1069
Jenkins, Marcelo 28
Jomier, Geneviève 250
Juárez, Eliseo 751
Kelner, Judith 907
Kummerfeld, Bob..... 395
Ladeira, Marcelo 894
Leiss, Ernst 383, 613, 785, 797
Liñán, José L. 127
Llamosa, Ricardo 967
Loh, Stanley 75
López, Rodrigo 529
Losavio, Francis 38, 821, 845
Luengo, Vanda 941
Maffioli, Francesco 359
Magalhães, Leo..... 922

Mareco, Carlos 1012
Márquez, Luis E. 347
Martínez,Pablo 310
Martínez, Rafael 482
Martins, Ricardo 907
Masson, Favio 577
Matteo, Alfredo 38, 821, 845
Mauco, Ma. Virginia 979
Mejía, Marcelo 191, 1080
Mejía, Antonio 191
Meléndez, Iraida 250
Millán, Marta 177
Millán, Yrieslis..... 808
Mishra, Prateek 87
Monteiro, Fred 152
Montilva,Jonás 870
Mora, Javier 967
Morales, Jorge 760
Moreno, Rodrigo 772
Moura, Ana Maria 139
Muñoz, César 300
Nava, Alberto 395
Navarro, Gonzalo 671, 1125
Ordaz, Oscar 347
Orozco, Javier 1092, 1101
Orth, Afonso 14
Ortiz, Martha 442
Ortiz, Carlos 589
Otsuka, Joice Lee 894
Ovalle, Demetrio..... 1021
Paccanaro, Alberto 1012, 1069
Pámanes, Alfonso 751
Pavel, Paulo 647
Pérez, Patricio 1033
Pons, Claudia 857
Queiruga, Claudia 637
Quintero, Alejandro 115, 506
Rabelo Jr., Alvaro 647
Ramos, Fernando 127, 482
Ramos, Rodrigo 715
Raposo, Alberto 922
Rivas, Rachele 808
Rodríguez, Rosseline 553
Rojas, Patricio 51
Rueda, Camilo 929
Rueda, Francisco 1113
Rukoz, Marta 250
Sadok, Djamel 907
Salas, Jorge 288
Salgado, Ana C. 625
Sánchez, Arturo..... 541
Sánchez, Marcelo 589
Santana, Regina 417
Santana, Marcos 417
Santos, Jorge 577, 1092, 1101
Santos, Marizete 625
Santos, Rodrigo 577
Savino, Nunzio 214, 371
Schneider, Ruben 276
Schwertner, Andréa 739
Silveira, Ricardo 894
Smith, Clara 165
Soares, Breno 152
Solar, Mauricio 1033
Sturm, Carolina 75
Suárez, Ascánder 288, 517
Tamura, Gabriel 529
Tau, Carlos 165
Tavares, Beatriz 1004
Tineo, Leodin 553
Toranzo, Marco 152
Trefftz, Christian 683
Ucrós, Ma. Eugenia 506
Valdeni, José 494, 894
Valencia, Frank 929
Vasconcelos, Alexandre 152
Villalón, Ricardo 28
Visconti, Marcello 51, 589, 953
von Brand, Horst 702
Wainer, Gabriel 565, 602
Wang, Yili..... 383
Watkins, Francisco 1033
Werner, Cláudia 833
Xexéo, Geraldo 63
You, Jianyu 785, 797
Zambón, Marcelo 577

Impreso por:
Centro de Publicaciones
Universidad de los Andes
Tel.: 281 58 24
Conmutador: 286 92 11 Ext. 2167
Santafé de Bogotá



Computación Avanzada en Ingeniería

Universidad de los Andes
Santafé de Bogotá

Investigación y Desarrollo
Consultoría Especializada

http://wwwing.uniandes.edu.co
e-mail : mox@uniandes.edu.co

Cra. 1 Este N° 18A-10 Santafé de Bogotá - Colombia
Tel : (57)(1) 2841043 - (57)(1) 2849911 ext 2814 - 2800 Fax : (57)(1) 2811518

Patrocinios

Cray Research - Colcomp S. A.
Dicolsis
Digital - STA
Horizonte
NASCO S. A.
NCR
Oracle
Silicon Graphics - Procálculo
Sun Microsystems
Unisys

Apoyo

Banco de la República
Embajada de Francia
ICETEX
UNESCO

Promoción y realización

ACIS
Universidad de los Andes

ACIS

CLEI
CENTRO LATINOAMERICANO DE
ESTUDIOS EN INFORMATICA

universidad de los andes